CPE Software Suite

10G PON Chipset

for
10G PON Chipset PRX120
10G PON Chipset PRX126
10G PON Chipset PRX321
AnyWAN™ SoC URX851
AnyWAN™ SoC MxL25641

Programmer's Guide

MaxLinear Confidential

Revision 2.6, 2024-05-08
Reference ID 617357

Legal Notice

The content of this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by MaxLinear, Inc. MaxLinear, Inc. assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this document. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced into, stored in, or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of MaxLinear, Inc.

EXCEPT AS OTHERWISE PROVIDED EXPRESSLY IN WRITING BY MAXLINEAR, AND TO THE MAXIMUM EXTENT PERMITTED BY LAW: (A) THE MAXLINEAR PRODUCTS ARE PROVIDED ON AN “AS IS” BASIS WITHOUT REPRESENTATIONS OR WARRANTIES OF ANY KIND, INCLUDING WITHOUT LIMITATION ANY IMPLIED OR STATUTORY WARRANTIES AND ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; AND (B) MAXLINEAR DOES NOT GUARANTEE THAT THE PRODUCTS WILL BE FREE OF ERRORS OR DEFECTS. MAXLINEAR PRODUCTS SHOULD NOT BE USED IN ANY EMERGENCY, SECURITY, MILITARY, LIFE-SAVING, OR OTHER CRITICAL USE CASE WHERE A FAILURE OR MALFUNCTION COULD CAUSE PERSONAL INJURY OR DEATH, OR DAMAGE TO OR LOSS OF PROPERTY. USERS ASSUME ALL RISK FOR USING THE MAXLINEAR PRODUCTS IN SUCH USE CASE. CUSTOMERS AND USERS ARE SOLELY RESPONSIBLE FOR USING THEIR OWN SKILL AND JUDGMENT TO DETERMINE WHETHER MAXLINEAR PRODUCTS ARE SUITABLE FOR THE INTENDED USE CASE.

MaxLinear, Inc. may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from MaxLinear, Inc., the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

MaxLinear, the MaxLinear logo, any other MaxLinear trademarks (including but not limited to MxL, Full-Spectrum Capture, FSC, AirPHY, Puma, AnyWAN, VectorBoost, MXL WARE, and Panther), are all property of MaxLinear, Inc. or one of MaxLinear's subsidiaries in the U.S.A. and other countries. All rights reserved.

All third-party marks and logos are trademarks™ or registered® trademarks of their respective holders/owners. Use of such marks does not imply any affiliation with, sponsorship or endorsement by the owners/holders of such trademarks. All references by MaxLinear to third party trademarks are intended to constitute nominative fair use under applicable trademark laws.

The URLs provided are for informational purposes only; they do not constitute an endorsement or an approval by MaxLinear of any of the products or services of the corporation or organization or individual. MaxLinear bears no responsibility for the accuracy, legality or content of the external site or for that of subsequent links. Contact the external site for answers to questions regarding its content.

MaxLinear, Inc.
5966 La Place Court, Suite 100
Carlsbad, CA 92008
Tel.: +1 (760) 692-0711
Fax: +1 (760) 444-8598
www.maxlinear.com

Revision History

Current:	**Revision 2.6, 2024-05-08**
Previous:	**Revision 2.5, 2023-09-08**

Page	Major changes since previous revision
All	The description refers to the 10G PON Chipset System Package release **1.22.0** or later.
All	Minor language improvements and typographical corrections
23	**Figure 1**, **PON-specific Software Modules** was updated.
40	**Chapter 2.4.3.3**, **UCI Configuration – ITU TDM PON**, **Table 3**, **PON TC Layer Interoperability Settings** was amended.
49	**Chapter 2.4.3.7**, **UCI Configuration – OMCI**, **Table 5**, **OMCI Configuration** was updated.
52	**Chapter 2.4.3.7.2**, **OMCI Interoperability Settings**, **Table 6**, **OMCI Interoperability Settings**, the interoperability option 8 was added.
55	**Chapter 2.4.3.8**, **UCI Configuration – Optical Interface**, **Table 7**, the description of `tx_en_mode`, `opt_tx_sd_polarity`, and `tx_pup_mode` was corrected. The description of `wl_switch_delay` was added.
72	**Chapter 2.4.4.3**, **SerDes Settings in the Device Tree**, the example was updated.
90	**Chapter 2.8**, **Under-voltage Detection** was added.
91	**Chapter 2.9**, **Interrupt Handling** was added.
172	**Chapter 4.1.1**, **OMCC Versions**, **Table 27**, **OMCC Version Numbering** was enhanced and information regarding padding was detailed.
185	**Chapter 4.3**, **Managed Entity Overview**, **Table 28**, **Managed Entity Implementation Overview** was updated.
204	**Chapter 4.8**, **Table 35**, **OMCI Alarm Overview** was updated
224	**Chapter 4.20**, **OMCI Managed Entity Implementation** was enhanced.
271	**Chapter 4.20.2.2**, **ANI2-G** was added.
292	**Chapter 4.20.2.10**, **FEC Performance Monitoring History Data** was updated, the managed entity can also be related to a TWDM channel in NG-PON2 applications.
309	**Chapter 4.20.2.13.1**, **Traffic Policer Implementation**, the maximum burst size limit was added.
311	**Chapter 4.20.2.14**, **GEM Port Network CTP Performance Monitoring History Data** was added.
346	**Chapter 4.20.3.14**, **Extended VLAN Tagging Operation Configuration Data**, the description was updated.
410	**Chapter 4.20.4.2**, **IPv6 Host Config Data** to **Chapter 4.20.4.5**, **TCP/UDP Performance Monitoring History Data** were added.
418	**Chapter 4.20.5.1**, **PPTP Ethernet UNI**, **Ethernet Loopback Configuration**, details were added.
424	**Chapter 4.20.5.3**, **Power over Ethernet Control** to **Chapter 4.20.5.10**, **Ethernet Frame Extended Performance Monitoring History Data 64-bit** were added.
461	**Chapter 4.20.6.4**, **Authentication Security Method** was added.
463	**Chapter 4.20.6.6**, **Octet String** was added.
469	**Chapter 4.20.6.12**, **Attribute** was added.
471	**Chapter 4.20.6.13**, **Openflow Configuration Data** was added.
473	**Chapter 4.20.6.14**, **General Purpose Buffer** was added.
474	**Chapter 4.20.6.15**, **File Transfer Controller** was added.

Revision History

Current:	**Revision 2.6, 2024-05-08**
Previous:	**Revision 2.5, 2023-09-08**
478	**Chapter 4.20.7.1**, **PPTP LCT UNI** was updated.
489	**Chapter 4.20.9**, **XG-PON Enhancements** was added.
496	**Chapter 4.20.10**, **XGS-PON Enhancements** was added.
496	**Chapter 4.20.10.1**, **Enhanced FEC Performance Monitoring History Data** was added.
498	**Chapter 4.20.10.2**, **Enhanced TC Performance Monitoring History Data** was added.
512	**Chapter 4.20.11.5**, **TWDM Channel PLOAM Performance Monitoring History Data Part 1** to **Chapter 4.20.11.11**, **Chapter 4.20.11.11** was added.
527	**Chapter 4.20.12**, **Vendor-specific OMCI Enhancements for Verizon** was added.
561	**Chapter 9.1.3**, **PHY Master Mode Operation** was added.
610	**Chapter 14.1.2**, **Disable the UART0 Access** was added.
610	**Chapter 14.1.3**, **Enable ToD Output on UART0** was added.
616	**Chapter 14.3.3**, **PON Command Line Functions**, the function `debug_test_pattern_enable` was enhanced by a wavelength selection parameter.
628	**Chapter 14.3.3.11**, **PON CLI – Dying Gasp Check** was added.
636	**Chapter 14.3.6**, **PON Debugging with pond** was enhanced by a comment.

Third-Party Trademark and Registered Trademark Declarations

Adtran®	Adtran is a registered trademark of Adtran Inc., registered in the U.S. and other countries and regions.
Alcatel®	Alcatel is a registered trademark of Nokia Corporation.
Apple®	Apple is a registered trademark of Apple Inc., registered in the U.S. and other countries and regions.
AppleTalk®	Apple is a registered trademark of Apple Inc., registered in the U.S. and other countries and regions.
Aptiv®	Aptiv is a registered trademark of Aptiv Technologies Ltd.
Ciena®	Ciena is a registered trademark of Ciena Corporation.
DZS®	DZS is a registered trademark of Dasan Zhone Solutions, Inc.
Git™	Git and the Git logo are either registered trademarks or trademarks of Software Freedom Conservancy, Inc., corporate home of the Git Project, in the United States and/or other countries.
GitHub®	GITHUB®, the GITHUB® logo design, the INVERTOCAT logo design, OCTOCAT®, and the OCTOCAT® logo design are trademarks of GitHub, Inc., registered in the United States and other countries.
GitLab®	GIT is a trademark of Software Freedom Conservancy and the use of "GitLab" is under license.
GNU®	GNU is a registered trademark of the Free Software Foundation.
Hisense®	Hisense is a registered trademark of Hisense International Co., Ltd.
HUAWEI®	HUAWEI is a registered trademark of Huawei Technologies Co., Ltd.
IEEE®	IEEE, the IEEE logo, and other IEEE logos and titles are registered trademarks or service marks of The Institute of Electrical and Electronics Engineers, Incorporated.
IETF®	IETF is a trademark of IETF Trust, registered in the U.S. and other countries.
Intel®	Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries.
ISO®	ISO and the ISO logo are registered trademarks of the International Organization for Standardization.

ITU-T®	ITU-T is a registered trademark of the ITU, a specialized agency of the United Nations.
JavaScript®	Registered trademark of Oracle in the United States.
Linux®	Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.
MACOM®	MACOM, M/A-COM, MACOM Technology Solutions, MACOM Tech, Partners in RF & Microwave, Partners from RF to Light and related logos are trademarks of MACOM Technology Solutions Holdings, Inc.
MIPS®	MIPS and the MIPS logo are trademarks or registered trademarks of MIPS Tech LLC in the United States and other countries.
MoCA®	MoCA is a registered trademarks of Multimedia over Coax Alliance in the United States and other countries.
Nokia®	Nokia is a registered trademark of Nokia Corporation.
NIST®	NIST is a registered trademark of the National Institute of Standards and Technology.
NXP®	NXP is a trademark of NXP B.V.
OpenSSL®	OpenSSL is a registered trademark owned by OpenSSL Software Foundation.
OpenWrt®	OpenWrt is a registered trademark owned by Software in the Public Interest, Inc.
PCI Express®	PCI-SIG®, PCIe®, and PCI Express® are registered trademarks and/or service marks of PCI-SIG.
PCIe®	PCI-SIG®, PCIE®, and PCI Express® are registered trademarks and/or service marks of PCI-SIG.
Python®	Python and PyCon are trademarks or registered trademarks of the Python Software Foundation.
RDK®	RDK is a registered trademark of RDK Management, LLC.
The Linux Foundation®	The Linux Foundation® is a trademark of The Linux Foundation in the United States and/or other countries.
Tibit®	Tibit is a registered trademark of Tibit Communications, Inc. (meanwhile aquired by Ciena).

UNIX®	UNIX is a registered trademark of The Open Group in the United States and other countries.
Verizon®	Verizon is a registered trademark of Verizon Communications Inc.
Wireshark®	Wireshark is a registered trademark of Sysdig, Inc.
Yocto Project®	Yocto Project® is a trademark of The Linux Foundation in the United States and/or other countries.

All other trademarks or registered trademarks are the property of their respective holders/owners.

Table of Contents

List of Figures

List of Tables

Preface

This document describes the software handling to implement the management software of a PON ONU system. The software supports the OMCI [56] standard.

For ONU devices in an SFP form factor, a software package is provided to implement the digital diagnostic monitoring interface (DDMI).

For applications which must support synchronous timing, the implementation of time of day (ToD) and the precision timing protocol (PTP, IEEE 1588) are described.

It is also possible to use the software package to implement Active Optical Ethernet (AON) fiber-based WAN interface applications.

Document Conventions

In the interest of brevity, this document uses short names to represent full MaxLinear product names.

UGW Software	Universal Gateway Software
PRX120	10G PON Chipset PRX120
PRX126	10G PON Chipset PRX126
PRX321	10G PON Chipset PRX321
10G PON Chipset, PRX300	PRX120, PRX126, and PRX321 devices
URX85x/MxL256xx	AnyWAN™ Broadband SoCs URX85x/MxL256xx

Software Release Information

For each software release, there is a dedicated system package release note [9], check for details related to software version numbering and open issues.

Releases (software development kits) are numbered 10G PON Chipset System Package x.y.z where:

- x = Major version
  - This increments with major functional upgrades or Linux kernel changes.
- y = Minor version
  - This increments with each full 10G PON Chipset System Package release, API changes are possible.
- z = Patch version
  - This increments with each patch release based on a full 10G PON Chipset System Package release, user APIs do not change.

Generated Documentation

The description of PON-related functions and data structures is provided within the source code, using Doxygen-formatted comments. Reference documentation is created by using the Doxygen tool [25]. Use the Doxygen configuration files provided within the library sources under `<library_name>/doc/doxyconfig` to create html files. It is possible to create other output formats, when the configuration file is adapted.

For generating other references, refer to the UGW Software documentation of the generation flow in chapter 1.1 of [19].

Other Documentation

For other references, refer to the UGW documentation [19] available on the MaxLinear myMxL website at **www.maxlinear.com/mymxl**.

For references to OpenWrt, refer to the online documentation available under **https://openwrt.org/**.

Document Information

This document's identifying information is:

Document Reference (Citation) Name

10G PON Subsystem Programmer's Guide Rev. 2.6

File Name

`617357_10G_PON_PG_Rev2.6.pdf`

Organization of this Document

- **Chapter 1**, **Introduction**
  General introduction to the software handling
- **Chapter 2**, **General Programming**
  PON library structure and general usage
- **Chapter 3**, **Networking Configuration**
  General usage of the networking functions
- **Chapter 4**, **OMCI Management Stack**
  OMCI management stack implementation
- **Chapter 5**, **PLOAM Protocol**
  PLOAM management protocol handling
- **Chapter 6**, **PON Library**
  PON functional application programming interface
- **Chapter 7**, **PON Image Library**
  PON software image upgrade programming interface
- **Chapter 8**, **PON Adapter**
  PON adapter structure and usage
- **Chapter 9**, **Frequency and Time Synchronization**
  Synchronous Ethernet, PTP, and ToD implementation
- **Chapter 10**, **Optical Diagnostics Monitoring**
  DDMI implementation
- **Chapter 11**, **Home Gateway Units**
  Implementation differences for HGU devices
- **Chapter 12**, **RDK-B System Differences**
  Implementation differences for HGU devices
- **Chapter 13**, **prplOS System Differences**
  Implementation differences for HGU devices
- **Chapter 14**, **Debug Functions**
  Debug functions
- **Chapter 15**, **Security Functions**
  Software security aspects
- **Literature References** and **Standards References**
  Related documents for further information
- **Terminology**
  Abbreviations and acronyms

1 Introduction

The PON-specific software modules enable the implementation of a PON ONU based on the 10G PON Chipset devices for SFU/SFP models and the AnyWAN™ Broadband SoCs URX85x/MxL256xx devices for HGU models as listed on the cover page. These additional modules are used in the larger scope of the UGW Software development environment (CPE Software Suite). This document covers these PON-specific modules:

- OMCI management stack (`gpon_omci_onu`)
- PON management adapter (`pon_adapter`)
- PON library (`pon_lib`)
- Ethernet clock synchronization
- PTP timing functions
- ToD timing functions
- DDMI optical monitoring functions
- Debug functions
- Security functions

Figure 1 depicts the relationship of the software parts. Refer to **[8]** for a detailed overview. When not mentioned, this document reflects the functionality 10G PON Chipset System Package 1.22 or later.

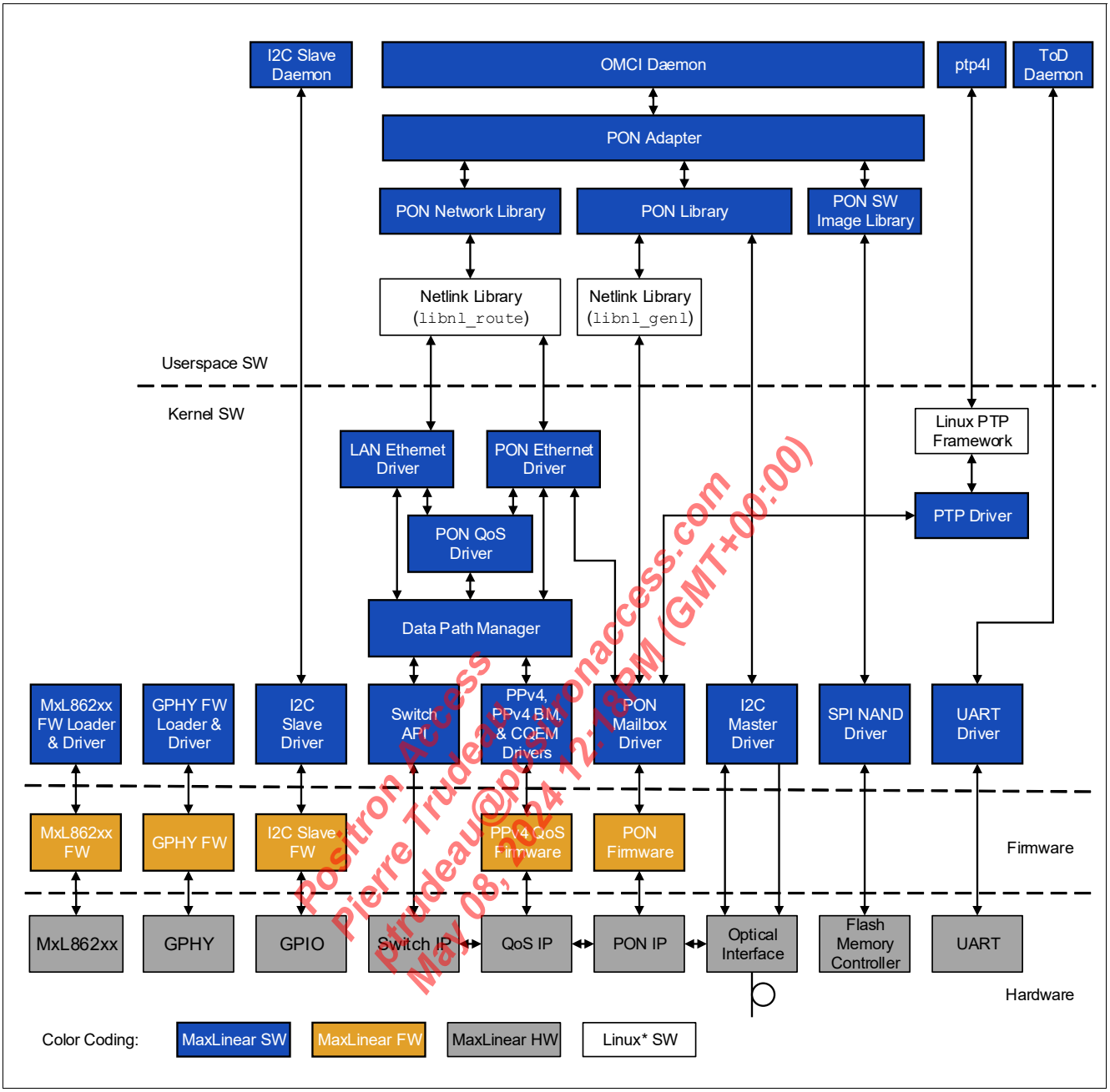


Figure 1 PON-specific Software Modules

1.1 10G PON Chipset System Package Installation

The 10G PON Chipset System Package (or Software Development Kit, SDK) is delivered as a compressed file, for example named `UGW-8.x_master-10GPON-1.22.0-SW-CD.zip`. Complete these steps to get started:

1. Uncompress the file `unzip UGW-8.x_master-10GPON-1.22.0-SW-CD.zip`.
2. Extract the installer.
   a) `./install.sh`
   b) Read the license agreement. Press the Enter or space bar key to move the text down. Type *yes* to accept the license agreement. The installer extracts the `ugw_sw` folder with sources.
3. Enter the source directory `cd ugw_sw`.
4. Prepare the source files.
   a) `./ugw-prepare-all.sh`
   This script prepares source code files.
   This step requires an active Internet connection and GIT to be installed and configured as it creates a GIT clone and downloads various packages from the Internet.
   It generates the `openwrt` folder which is the top level for the software builds.
5. Enter the working directory `cd openwrt`.
6. Select a model by running.
   a) `./scripts/ltq_change_environment.sh switch`
   Choose one model from the list by entering the index number and press the Enter key.
   To create a fully capable file system with debug function enabled for all reference platforms, select `PRX300_DEBUG` to generate all debug-capable target images or a single one of the listed target image types (that fits the target hardware).
7. Build the software binary by using the `make` command.
   Speed up with the parallel option -j, for example: `make -j16`
   For verbose mode, use `make V=s`

These steps are also listed in the README file in the 10G PON Chipset System Package root folder.

The created binary files are located under `ugw_sw/openwrt/bin/targets/intel_mips/prx300`.

Amongst the different types of images created, the images used in U-Boot for updating the complete flash are named `lede-intel_mips-prx300-PRX<..>-squashfs-fullimage.img`.

1.1.1 SDK Version Scheme

The SDK versions are numbered as x.y.z with

- x: Major version, increased with major Linux kernel changes
- y: Minor version, increased with each regular release
- z: Patch version, increased for patch versions that are based on a regular release.

1.2 Software Licenses

The software modules contained in the 10G PON Chipset System Package are provided under individual licenses. For further information, refer to [21].

To list the licenses from the local software installation, use this command from the `/openwrt` directory:

```
PKGS=$(echo $(sed -n -e
'/CONFIG_PACKAGE_.*=y/ s/CONFIG_PACKAGE_\(.*\)=y/\1:/p' .config))&& ./scripts/pack
age-metadata.pl license tmp/.packageinfo | grep -E ${PKGS// /|}

argp-standalone-shared: Makefile.am
baresip-mod-dtmfio: BSD-3-Clause
baresip-mod-portaudio: BSD-3-Clause
baresip-mod-stdio: BSD-3-Clause
base-files: GPL-2.0
base-files-ieee1588: GPL-2.0-only
base-files-platform-prx300: GPL-2.0-only
bird1-ipv4-uci: GPL-3.0+
bird1-ipv6-uci: GPL-3.0+
busybox: GPL-2.0
cgi-io: GPL-2.0-or-later
dosfstools: GPL-3.0-or-later
dropbear: MIT
dtc: GPL-2.0
ethtool: GPL-2.0
fio: GPL-2.0+
firewall: ISC
freeradius3-mod-eap-gtc: GPL-2.0
freeswitch-stable-mod-portaudio: AGPL-3.0 Apache-2.0 BSD-2-Clause BSD-3-Clause BSD-
4-Clause BSD-like Beerware GPL-1.0+ GPL-2.0 GPL-2.0+ GPL-3.0 ISC LGPL-2.0+ LGPL-2.1
LGPL-2.1+ MIT/X11 (BSD like) MPL-1.1 OpenLDAP RSA-MD zlib-acknowledgement
freeswitch-stable-mod-rtc: AGPL-3.0 Apache-2.0 BSD-2-Clause BSD-3-Clause BSD-4-
Clause BSD-like Beerware GPL-1.0+ GPL-2.0 GPL-2.0+ GPL-3.0 ISC LGPL-2.0+ LGPL-2.1
LGPL-2.1+ MIT/X11 (BSD like) MPL-1.1 OpenLDAP RSA-MD zlib-acknowledgement
freeswitch-stable-mod-stress: AGPL-3.0 Apache-2.0 BSD-2-Clause BSD-3-Clause BSD-4-
Clause BSD-like Beerware GPL-1.0+ GPL-2.0 GPL-2.0+ GPL-3.0 ISC LGPL-2.0+ LGPL-2.1
LGPL-2.1+ MIT/X11 (BSD like) MPL-1.1 OpenLDAP RSA-MD zlib-acknowledgement
fstools: GPL-2.0
getrandom: GPL-2.0
gpon-libs: GPL-2.0-only
gpon-omci-onu: MaxLinear Commercial Use License
gpon-sfp-eeprom: GPL-2.0-only or BSD-2-Clause
gst1-mod-gio: LGPL-2.1-or-later GPL-2.0-or-later
gst1-mod-ivtc: LGPL-2.1-or-later GPL-2.0-or-later
gst1-mod-ossaudio: LGPL-2.1-or-later
ifxos: GPL-2.0-only BSD-2-Clause
ip-bridge: GPL-2.0
ip-tiny: GPL-2.0
jq: BSD
jshn: ISC
jsonfilter: ISC
kmod-dp-eth-reinsert: GPL-2.0
```

```
kmod-dp-oam: GPL-2.0
kmod-eeprom-at24: GPL-2.0
kmod-hwmon-core: GPL-2.0
kmod-i2c-algo-bit: GPL-2.0
kmod-i2c-core: GPL-2.0
kmod-i2c-dev: GPL-2.0
kmod-i2c-gpio: GPL-2.0
kmod-i2c-mux: GPL-2.0
kmod-i2c-mux-gpio: GPL-2.0
kmod-i2c-mux-pca954x: GPL-2.0
kmod-ifxos: GPL-2.0-only BSD-2-Clause
kmod-intel_eth_drv_xrx500: GPL-2.0
kmod-intel_eth_xrx500_fw: GPL-2.0
kmod-intel_ppv4_qos_drv: GPL-2.0
kmod-ipt-conntrack: GPL-2.0
kmod-ipt-core: GPL-2.0
kmod-ipt-nat: GPL-2.0
kmod-ipt-raw: GPL-2.0
kmod-leds-gpio: GPL-2.0
kmod-ledtrig-gpio: GPL-2.0
kmod-lib-cli: GPL-2.0-only BSD-2-Clause
kmod-lib-textsearch: GPL-2.0
kmod-mdio: GPL-2.0
kmod-mdio-gpio: GPL-2.0
kmod-nf-conntrack: GPL-2.0
kmod-nf-conntrack6: GPL-2.0
kmod-nf-ipt: GPL-2.0
kmod-nf-nat: GPL-2.0
kmod-nf-nathelper-extra: GPL-2.0
kmod-nf-reject: GPL-2.0
kmod-nvmem: GPL-2.0
kmod-of-mdio: GPL-2.0
kmod-pon-arp-ndp-reinsert: MaxLinear Commercial Use License
kmod-pon-mbox-drv: GPL-2.0-only or BSD-2-Clause
kmod-pon-mbox-drv-eth: GPL-2.0-only or BSD-2-Clause
kmod-pon-mbox-drv-ptp: GPL-2.0-only or BSD-2-Clause
kmod-pon-mcc-drv: GPL-2.0-only or BSD-2-Clause
kmod-pon-mcc-drv-chk: GPL-2.0-only or BSD-2-Clause
kmod-pon-mcc-drv-filter: GPL-2.0-only or BSD-2-Clause
kmod-pon-sfp-i2c: GPL-2.0-only or BSD-2-Clause
kmod-pps: GPL-2.0
kmod-pps-gpio: GPL-2.0
kmod-ptp: GPL-2.0
kmod-qos-tc: GPL-2.0
kmod-regmap-mmio: GPL-2.0
kmod-sched-act-colmark: GPL-2.0
kmod-sched-act-police: GPL-2.0
kmod-sched-act-vlan: GPL-2.0
kmod-sched-core: GPL-2.0
kmod-sched-drr: GPL-2.0
kmod-sched-flower: GPL-2.0
```

```
kmod-sched-mqprio: GPL-2.0
kmod-sched-prio: GPL-2.0
kmod-sched-red: GPL-2.0
kmod-serial-lantiq: GPL-2.0
kmod-sound-cs5535audio: GPL-2.0
kmod-spi-bitbang: GPL-2.0
kmod-spi-gpio: GPL-2.0
kmod-spi-intel_mips: GPL-2.0
kmod-spi-intel_mips-grx500: GPL-2.0
kmod-usb-audio: GPL-2.0
kmod-voice-kpi2udp: GPL-2.0-only or BSD-2-Clause
kmod-voice-sdd-mbx: GPL-2.0-only or BSD-2-Clause
kmod-voice-tapi: GPL-2.0-only or BSD-2-Clause
kmod-voice-vmmc-xrx500: GPL-2.0-only or BSD-2-Clause
kmod-voice-vmmc-xrx500-slic200: GPL-2.0-only or BSD-2-Clause
kmod-w1-master-gpio: GPL-2.0
lib-cli: GPL-2.0-only BSD-2-Clause
libaio: LGPL-2.1-only
libblobmsg-json: ISC
libc: GPL-3.0-with-GCC-exception
libgcc: GPL-3.0-with-GCC-exception
libgst1audio: LGPL-2.1-or-later GPL-2.0-or-later
libiio: LGPL-2.1
libip4tc: GPL-2.0
libip6tc: GPL-2.0
libiptc: GPL-2.0
libjson-c: MIT
libjson-script: ISC
libnl-core: LGPL-2.1
libnl-genl: LGPL-2.1
libnl-tiny: LGPL-2.1
libopenssl: OpenSSL
libpcap: BSD-3-Clause
libpthread: GPL-3.0-with-GCC-exception
librt: GPL-3.0-with-GCC-exception
libubox: ISC
libubus: LGPL-2.1
libuci: LGPL-2.1
libuclient: ISC
libugpio: LGPL-2.1-or-later
libxtables: GPL-2.0
linuxptp: GPL-2.0
logd: GPL-2.0
luci: Apache-2.0
luci-app-firewall: Apache-2.0
luci-lib-nixio: Apache-2.0
netifd: GPL-2.0
nginx-mod-luci: 2-clause BSD-like license
node-cylon-gpio: Apache-2.0
open-plc-utils-int6kmdio: ISC
opkg: GPL-2.0
```

```
perlbase-io: GPL-1.0-or-later Artistic-1.0-Perl
perlbase-perlio: GPL-1.0-or-later Artistic-1.0-Perl
php7-pecl-dio: PHPv3.01
pm-util: GPLv2
pon-adapter: MaxLinear Commercial Use License
pon-base-files: GPL-2.0-only
pon-base-files-debug: GPL-2.0-only
pon-base-files-sfu: GPL-2.0-only
pon-gpon-firmware-b: MAXLINEAR PROPRIETARY SOFTWARE LICENSE AGREEMENT
pon-img-lib: MaxLinear
pon-lib: MaxLinear Commercial Use License
pon-lib-cli: MaxLinear Commercial Use License
pon-lib-daemon: MaxLinear Commercial Use License
pon-libnl-core: LGPL-2.1
pon-libnl-genl: LGPL-2.1
pon-libnl-route: LGPL-2.1
pon-mcc-cli: GPL-2.0-only or BSD-2-Clause
pon-mcc-lib: GPL-2.0-only or BSD-2-Clause
pon-net-lib: MaxLinear Commercial Use License
pon-tod: MaxLinear Commercial Use License
pon-tools: BSD-2-Clause
pon-xpon-firmware-b: MAXLINEAR PROPRIETARY SOFTWARE LICENSE AGREEMENT
ponmbox-cli: MaxLinear Commercial Use License
portaudio: MIT
ppv4-qos-firmware: Intel
procd: GPL-2.0
python3-asyncio: Python/2.0
rpcd: ISC
rpcd-mod-luci: Apache-2.0
ruby-etc: BSD-2-Clause
ruby-stringio: BSD-2-Clause
safeclibs3: CISCO SYSTEMS
sane-kodakaio: GPL-2.0 GPL-2.0-or-later
stress: GPL-2.0
strongswan-mod-uci: GPL-2.0-or-later
sudo: ISC
synce-tool: GPL-2.0-only or BSD-2-Clause
tc: GPL-2.0
tc-mod-iptables: GPL-2.0
tcpdump-mini: BSD-3-Clause
testptp: GPL-2.0-only
tio: GPL-2.0-or-later
ubi-utils: GPLv2
uboot-envtools: GPL-2.0 GPL-2.0+
uboot-intel-mips: GPL-2.0-or-later
uboot-prx120-sfu-256-qspi-nand-gphy: GPL-2.0-or-later
uboot-prx120-sfu-qspi-nand-gphy: GPL-2.0-or-later
uboot-prx126-eva-qspi-nand: GPL-2.0-or-later
uboot-prx126-sfp-qspi-nand: GPL-2.0-or-later
uboot-prx321-cpe: GPL-2.0-or-later
uboot-prx321-eva-qspi-nand: GPL-2.0-or-later
```

```
uboot-prx321-eva-spi-nand: GPL-2.0-or-later
uboot-prx321-sfu-256-qspi-nand: GPL-2.0-or-later
uboot-prx321-sfu-256-v1-10-nand: GPL-2.0-or-later
uboot-prx321-sfu-lab-nand: GPL-2.0-or-later
uboot-prx321-sfu-qspi-nand: GPL-2.0-or-later
uboot-prx321-sfu-v1-10-nand: GPL-2.0-or-later
ubox: GPL-2.0
ubus: LGPL-2.1
ubusd: LGPL-2.1
uci: LGPL-2.1
uclient-fetch: ISC
uhttpd-mod-ubus: ISC
ulogd: GPL-2.0-only
urandom-seed: GPL-2.0
urngd: GPL-2.0 BSD-3-Clause
yate-mod-ilbcwebrtc: GPL-2.0
```

1.3 Linux Kernel Roadmap

The current 10G PON Chipset System Package releases use these Linux kernel versions:

- 4.9 for the 10G PON Chipset - LTS, end-of-life January 2023.
  Starting in second half of 2024, releases will use the Linux Kernel version 5.15, projected end-of-life October 2026
- 4.19 for the URX85x/MxL256xx - LTS, projected end-of-life December 2024
  Starting in 2023 the releases will use the Linux kernel version 5.15, projected end-of-life October 2026.

MaxLinear will only support new features in the latest Linux kernel version.

2 General Programming

This chapter provides general information about programming of PON applications and implementation hints.

2.1 Operating System

The software is based on the UGW Software distribution which itself builds on LEDE/OpenWrt. See **Section 14.3.1** for the versions.

2.1.1 Boot Loader

The boot process is built on several steps:

- Boot ROM processing
  The initial boot code is executed from an on-chip ROM.
- U-Boot
  This is the boot loader that prepares the system to allow DRAM access and configures the memory locations for the operating system.
- Linux kernel boot
- System application boot

During the first system boot some hardware components are automatically detected and the related configuration is prepared. This step is skipped for subsequent boot cycles. For details see **Section 2.4.3.8**, **Optical Configuration Database** and **Section 2.4.4.4**, **PON SerDes Configuration Database**.

Note: Generating the RSA private key for the SSH server at the first boot of the system can take up to 30 seconds. This affects the first boot only and not the subsequent boot sequences.
If the SSH server is not needed by the application, MaxLinear recommends removing it from the system by deselecting CONFIG_PACKAGE_dropbear in the build configuration.

Note: By default the Quad SPI mode must be enabled. This is achieved by selecting the configuration as CONFIG_QSPI_QUAD_MODE=y in the config file under ./ugw_sw/openwrt//build_dir/target-mips_24kc+nomips16_musl_prx300_debug/linux-intel_mips_prx300/u-boot-2016.07/configs.

2.2 Software Compilation

The 10G PON Chipset System Package provides C libraries which must be compiled with an appropriate C compiler. When the customer-added software is written in C++, the related C header files must be included and the pre-compiled C libraries must be linked to the C++ code.

The 10G PON Chipset System Package uses and supports the **musl** C compiler.

Attention: Do not compile the C libraries with a C++ compiler.

2.3 Software Update

In ITU PON systems, the software update is under control of the Optical Line Terminal (OLT) and performed through the OMCI protocol. See **Section 4.20.1.9**, **Software Image** for details. Additional aspects are provided hereafter.

For a manual software update under laboratory conditions, see **Section 14.15.3**, **Manual Software Update**.

2.3.1 Roll-back Prevention

Software updates are sometimes required to solve known functional or security issues. In such cases, there is a mechanism in place to ensure that an older outdated revision is never loaded.

To avoid re-installation of a software revision beyond a given version, the roll-back prevention mechanism is provided.

For details, refer to the UGW secure boot documentation [16].

2.3.2 Flash Wear-out Protection

For details, refer to the related UGW documentation [21].

2.3.3 Image Signing

To ensure that only validated images are loaded and run, each software image is signed with a private key which is then checked against a public key stored in the on-chip OTP memory.

Image signing is optional. For details, refer to the related UGW documentation [16].

2.4 Configuration and Initialization

This section describes the mechanisms used to configure and initialize specific ONU applications.

2.4.1 Factory Reset

To start from a factory-fresh configuration, it is possible to perform a factory reset which reinitializes the internal settings to their original values.

The simplest way to do this is to call "`firstboot`" from the Linux command line.

2.4.2 Device Tree

The application specific static configurations are defined in the Linux device tree files (`.dts`) and device tree include files (`.dtsi`) located at `ugw_sw/feeds/ugw/targets/intel_mips/dts`.

Table 1 lists the files specific to the PON applications. For the general handling, refer to [27], [24].

For details of individual components, refer to the source code documentation in `./ugw_sw/source/linux/Documentation/devicetree/bindings`.

The device tree is built on a set of files which are able to include other files. For PON applications, Figure 2 shows the related file structure.

Table 1 Device Tree Files

File Name	Contents	Purpose
Device Tree Top Level Files		
`prx126-sfp-pon.dts`	Data interface configuration Flash memory interface configuration	SFP-specific configurations for model PRX126-SFP-PON
`prx120-sfu-qspi-pon.dts`	Memory, SPI,...	SFU-specific configurations for model PRX120-SFU-PON
`prx321-sfu-qspi-pon.dts`	Memory, SPI,...	SFU-specific configurations for model PRX321-SFU-PON
`prx321-sfu-qspi-512-pon.dts`	Memory, SPI,...	SFU-specific configurations for model PRX321-SFU-PON with 512 MB memory
`prx321-sfu-qspi-256-pon.dts`	Memory, SPI,...	SFU-specific configurations for model PRX321-SFU-PON with 256 MB memory
`prx321-sfu-qspi-eth.dts`	Memory, SPI,...	SFU-specific configurations for model PRX321-SFU-ETH

Table 1 Device Tree Files (cont'd)

File Name	Contents	Purpose
`prx321-sfu-v1-10-*.dts`	Memory, SPI,...	SFU-specific configurations for models that are intended to be run on an EASY PRX321 V1.10 reference board.
`prx300-bootcode.dts`	Model definition	Configuration for the secure boot core in productive mode
Device Tree Include Files		
`prx300.dtsi`	Common settings	Covers all settings that are common to the PRX application family
`prx126-sfp.dtsi`	LAN and WAN interface configuration	SFP applications
`prx321-sfu.dtsi`	LAN and WAN interface configuration	SFU applications
`prx321-sfu-qspi.dtsi`	LAN and WAN interface configuration	SFU applications
`prx300-bootcore.dtsi`	Boot core configuration	Boot core

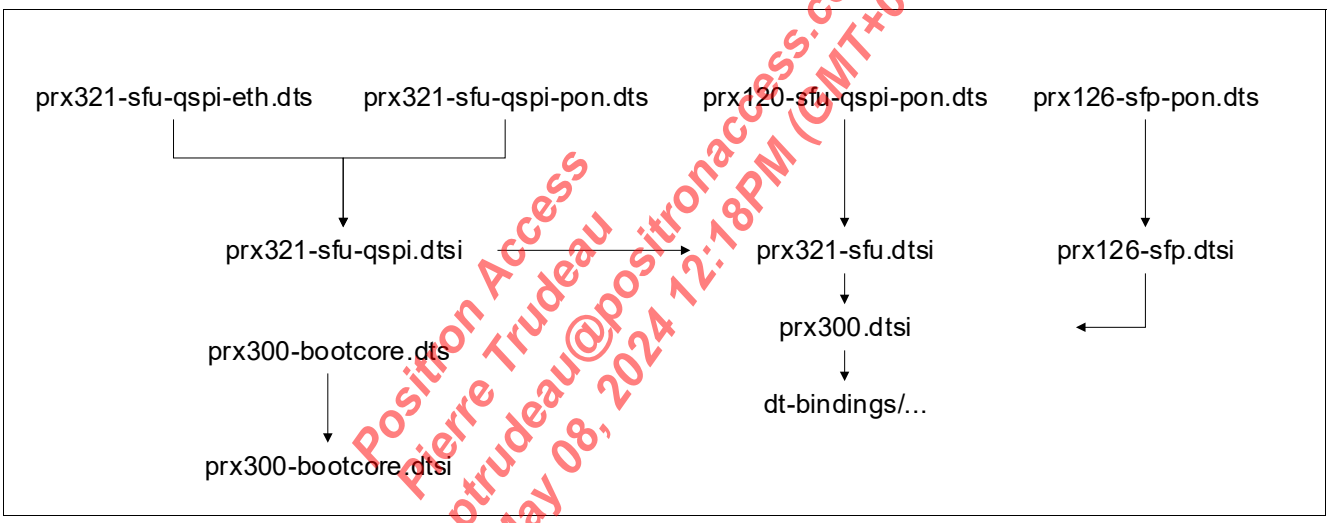


Figure 2 Device Tree Structure

While the `prx300.dtsi` contains SoC-related information that must not be changed, more specific files are adapted to certain applications. The contents depend on the application functionality and external circuitry connected to the specific device. These files must be adapted by the target ONU manufacturer, especially when the hardware platform differs from the reference designs.

The files related to the boot core must not be modified.

2.4.2.1 GPHY LED Device Tree Settings

The integrated GPHY device provides the option to drive up to three status LEDs directly from the hardware. This avoids the need of GPIO handling by the software. The LED output mode is enabled in the device tree while the specific LED function must be configured in the GPHY IP itself. Refer to the GPHY API documentation **[15]** and to the standalone GPHY documentation **[17]** for more details.

This is an example of the device tree setting in `prx321-sfu.dtsi`. The lines `intel,sso-hw-trigger;` enable direct LED control by the hardware. The GPIO software control is selected when these lines are removed.

```
led2: led2 {
        label = "led2:green:gphy";
        led-gpio = <&ssogpio 2 GPIO_ACTIVE_HIGH>;
        intel,led-pin = <2>;
        intel,sso-hw-trigger;
};

led3: led3 {
        label = "led2:red:gphy";
        led-gpio = <&ssogpio 3 GPIO_ACTIVE_HIGH>;
        intel,led-pin = <3>;
        intel,sso-hw-trigger;
};

led4: led4 {
        label = "led4:yellow:gphy";
        led-gpio = <&ssogpio 4 GPIO_ACTIVE_HIGH>;
        intel,led-pin = <4>;
        intel,sso-hw-trigger;
};
```

Note: The GPHY control functions are not part of the 10G PON Chipset System Package delivery and must be applied to the PHY driver in the kernel.

2.4.2.2 Enable Packet Padding

The packet processor has the capability to enable packet padding for short packets, for example when a VLAN tag is removed.

Note: If MACsec is enabled then packet padding cannot be used.

Enable this feature by using device tree properties. Several properties must be enabled in several drivers device tree nodes.

These properties must be changed for the correct functionality:

- Ethernet driver - *intel,with-rx-fcs*
- PON Ethernet driver - *intel,with-rx-fcs*
- Switch driver (gsw_core) - *intel,pmac-padding-fcs-mode = <MXL_PMAC_PADDING_MODE_EN>*
- MAC diver - *intel,rx-fcs-mode = <MXL_RX_FCS_NO_REMOVE> and intel,tx-fcs-rm = <MXL_TX_FCS_REMOVE>*

The current settings are available in *"/sys/firmware/devicetree/base*". Here is an example of such a change:

```
[...]
#include <dt-bindings/net/mxl,mac-drv.h>
#include <dt-bindings/net/mxl,switch-api-drv.h>
[...]

&eth {
        status = "okay";
       lan0: interface@0 {
              [...]
              intel,with-rx-fcs;
              [...]
        };
       [...]
};

&pon_eth {
        intel,with-rx-fcs;
};

&gsw_core {
       [...]
       intel,pmac-padding-fcs-mode = <MXL_PMAC_PADDING_MODE_EN>;
};

&gsw_mac1 {
        [...]
        intel,rx-fcs-mode = <MXL_RX_FCS_NO_REMOVE>;
        intel,tx-fcs-rm = <MXL_TX_FCS_REMOVE>;
};

&gsw_mac2 {
       [...]
       intel,rx-fcs-mode = <MXL_RX_FCS_NO_REMOVE>;
       intel,tx-fcs-rm = <MXL_TX_FCS_REMOVE>;
};
```

2.4.3 UCI Configuration

The multiple configuration items are held as UCI configuration, located at `/etc/config`.

These configuration files related to PON applications are available:

- `dropbear`
- `firewall`
- `gpon`
- `ieee_1588`
- `network`
- `omci`
- `optic`
- `ponnetwork`
- `serdes`
- `sfp_eeprom`
- `system`
- `tod`
- `todd`

On a secure system image, these configuration files are read-only and must be provisioned during ONU manufacturing. Modification to the UCI on a running system is only possible if the system image is debug-enabled.

For differences on HGU systems see **Chapter 11.5**.

2.4.3.1 UCI Configuration – Dropbear

These configuration values are used to establish a secure SSH connection.

For example:

```
# cat /etc/config/dropbear

config dropbear
        option PasswordAuth     'on'
        option RootPasswordAuth 'on'
        option Port             '22'
```

For details, refer to the Dropbear online resources.

2.4.3.2 UCI Configuration – Firewall

These configurations specify the data forwarding to the CPU. The application-specific adaptation is required. The default settings are:

```
# cat /etc/config/firewall

config defaults
        option syn_flood '1'
        option input 'ACCEPT'
        option output 'ACCEPT'
        option forward 'REJECT'

config zone
        option name 'lan'
        list network 'lan'
        list network 'lct'
        option input 'ACCEPT'
        option output 'ACCEPT'
```

```
        option forward 'ACCEPT'

config zone
        option name 'wan'
        list network 'wan'
        list network 'wan6'
        option input 'REJECT'
        option output 'ACCEPT'
        option forward 'REJECT'
        option masq '1'
        option mtu_fix '1'

config forwarding
        option src 'lan'
        option dest 'wan'

config rule
        option name 'Allow-DHCP-Renew'
        option src 'wan'
        option proto 'udp'
        option dest_port '68'
        option target 'ACCEPT'
        option family 'ipv4'

config rule
        option name 'Allow-Ping'
        option src 'wan'
        option proto 'icmp'
        option icmp_type 'echo-request'
        option family 'ipv4'
        option target 'ACCEPT'

config rule
        option name 'Allow-IGMP'
        option src 'wan'
        option proto 'igmp'
        option family 'ipv4'
        option target 'ACCEPT'

config rule
        option name 'Allow-DHCPv6'
        option src 'wan'
        option proto 'udp'
        option src_ip 'fc00::/6'
        option dest_ip 'fc00::/6'
        option dest_port '546'
        option family 'ipv6'
        option target 'ACCEPT'

config rule
        option name 'Allow-MLD'
```

```
        option src 'wan'
        option proto 'icmp'
        option src_ip 'fe80::/10'
        list icmp_type '130/0'
        list icmp_type '131/0'
        list icmp_type '132/0'
        list icmp_type '143/0'
        option family 'ipv6'
        option target 'ACCEPT'

config rule
        option name 'Allow-ICMPv6-Input'
        option src 'wan'
        option proto 'icmp'
        list icmp_type 'echo-request'
        list icmp_type 'echo-reply'
        list icmp_type 'destination-unreachable'
        list icmp_type 'packet-too-big'
        list icmp_type 'time-exceeded'
        list icmp_type 'bad-header'
        list icmp_type 'unknown-header-type'
        list icmp_type 'router-solicitation'
        list icmp_type 'neighbour-solicitation'
        list icmp_type 'router-advertisement'
        list icmp_type 'neighbour-advertisement'
        option limit '1000/sec'
        option family 'ipv6'
        option target 'ACCEPT'

config rule
        option name 'Allow-ICMPv6-Forward'
        option src 'wan'
        option dest '*'
        option proto 'icmp'
        list icmp_type 'echo-request'
        list icmp_type 'echo-reply'
        list icmp_type 'destination-unreachable'
        list icmp_type 'packet-too-big'
        list icmp_type 'time-exceeded'
        list icmp_type 'bad-header'
        list icmp_type 'unknown-header-type'
        option limit '1000/sec'
        option family 'ipv6'
        option target 'ACCEPT'

config rule
        option name 'Allow-IPSec-ESP'
        option src 'wan'
        option dest 'lan'
        option proto 'esp'
        option target 'ACCEPT'
```

```
config rule
        option name 'Allow-ISAKMP'
        option src 'wan'
        option dest 'lan'
        option dest_port '500'
        option proto 'udp'
        option target 'ACCEPT'

config include
        option path '/etc/firewall.user'

config rule
        option name 'Allow-SSH'
        option src 'lan'
        option proto 'tcpudp'
        option dest_port '22'
        option target 'ACCEPT'

config rule
        option name 'Default-Block-WAN'
        option src 'wan'
        option proto 'tcpudp'
        option target 'REJECT'

config rule
        option name 'Default-Block-LAN'
        option src 'lan'
        option proto 'tcpudp'
        option target 'REJECT'
```

For details, refer to the Linux documentation.

2.4.3.3 UCI Configuration – ITU TDM PON

These configuration values are only used when one of the single-wavelength ITU-T Time Division Multiplex (TDM) PON operation modes (G-PON, XG-PON, or XGS-PON) or the Active Optical Ethernet (AON) mode is selected.

For example:

```
# cat /etc/config/gpon

config gpon 'ploam'
        option nSerial 'MAXL12345678'
        option regID '0x49 0x4e 0x54 0x45 0x4c 0x20 0x46 0x41 0x4c 0x43 0x4f 0x4e
0x20 0x4d 0x4f 0x55 0x4e 0x54 0x41 0x49 0x4e 0x20 0x4c 0x51 0x44 0x45 0xF1 0x50 0xED
0xF3 0x20 0x20 0x20 0x20 0x20 0x20'
        option ploam_timeout_0 '65535'
        option ploam_timeout_1 '10000'
        option ploam_timeout_2   '125'
        option ploam_timeout_6 '10000'

config gpon 'ponip'
        option pon_mode 'xgspon'
```

```
option iop_mask '18'
```

Note: To switch the operation mode between one of the PON modes or the AON mode, the device must be re-booted, either with a different device tree (if selectable in U-Boot) or a different software image. The 10G PON Chipset System Package creates a dedicated AON (“eth”) image.

PON Operation Mode Settings

Table 2 PON Operation Mode Configuration

Group	Option	Unit	Range	Default	Notes
ploam	nSerial	ASCII	Up to 12 characters	`MAXL12345678`	This is the serial number used by the Physical Layer OAM (PLOAM) protocol. The value is represented as a 12-byte ASCII string.
	regID	–	Up to 36 byte	`0x49 … 0x20`	36-byte registration ID used by the PLOAM registration protocol for XG-PON and XGS-PON. The value is presented as 36 individual bytes in hexadecimal notation to form an ASCII string. In G-PON operation mode, the rightmost (lower) ten bytes of the registration ID are used as the PLOAM password.
	timeout_0	ms	0 to 65535	`65535`	This timeout value (TOZ) is used to return from PLOAM state O23 in XG-PON and XGS-PON. When set to 0, the timeout is disabled. This value is not applicable to the G-PON operation mode.
	timeout_1	ms	0 to 65535	`10000`	This value configures the value of TO1. The value is used as ranging timeout (in PLOAM state O4). This value is applicable to all ITU PON operation modes.
	timeout_2	ms	0 to 65535	`125`	This value configures the value of TO2 (LODS re-initialization timer). Intermittent LODS timeout (in PLOAM state O6 for XG-PON and XGS-PON operation mode) or POPUP timeout (O6 in G-PON operation mode). This value is applicable to all ITU PON operation modes.
	timeout_6	ms	0 to 65535	`10000`	This value configures the value of TO6 for the Lost ONU timeout (triggers if no allocations are received in PLOAM state O5.x). If set to 0, the timeout is disabled.

Table 2 PON Operation Mode Configuration (cont'd)

Group	Option	Unit	Range	Default	Notes
ponip	pon_mode	–	–	`xgspon`	These are the supported TDM PON and active optical Ethernet (AON) operation modes: • `gpon` • `xgpon` • `xgspon` • `aon`
	iop_mask	–	See **Table 3**.	`18`	**Table 3** shows the available interoperability options for `iop_mask`. Each of the bits are individually set or cleared to enable or disable the related interoperability option.
authen-tication	psk	–	128 bit	0x0123456789ABCDEF	This is a unique 128-bit pre-shared key. *Note: Replace the trivial default for production.*
	mode	–	1	1	Encryption mode 0_D **NONE** No encryption support. 1_D **AES** Supports AES-CMAC-128. 2_D **SHA256** Supports HMAC-SHA-256. 2_D **SHA512** Supports HMAC-SHA-512. *Note: Only AES-CMAC-128 is supported.*
	key_size	–	128	128	Encryption key size. *Note: Only a key size of 128 bit is supported.*

Table 3 PON TC Layer Interoperability Settings

IOP Bit	Interoperability Function
0	Disable the OMCC encryption (upstream and downstream) when required for interoperability with some OLTs. 0_B **EN** OMCC encryption is enabled (default). 1_B **DIS** OMCC encryption is disabled.
1	Disable the OMCC encryption in upstream direction when required for interoperability with some OLTs. 0_B **EN** OMCC upstream encryption is enabled (default). 1_B **DIS** OMCC upstream encryption is disabled.
2	An additional ranging offset between downstream and upstream processing is inserted and added to the ranging time. This is required for specific OLT types. 0_B **OFF** The additional ranging offset is disabled (default). 1_B **ON** The additional ranging offset is enabled.
3	Reserved
4	Automatically enters the PLOAM state O7 (emergency stop) when a rogue ONU situation has been detected by the automatic rogue ONU detection mechanism. 0_B **OFF** The PLOAM state is not automatically changed (default). 1_B **ON** The PLOAM state is changed to O7 and stored into non-volatile memory.
5	This mode enables XG-PON operation with 10G upstream speed (instead of 2.5G) which is needed for specific OLT types. It is only allowed to set this bit when the operation mode is set to XG-PON (G.987). 0_B **OFF** Standard behavior (default). 1_B **ON** Vendor-specific behavior.

Table 3 PON TC Layer Interoperability Settings (cont'd)

IOP Bit	Interoperability Function
6	This mode handles a vendor-specific PLOAM downstream message with message ID = 0xFF. 0_B **OFF** The PLOAM message 0xFF is handled as unknown (default). 1_B **ON** Vendor-specific PLOAM message handling.
7	This mode enables BIP error counting while the downstream FEC mode is enabled. This must only be enabled when the OLT supports it, otherwise excessive BIP errors are counted and the SD/SF alarms are raised. 0_B **OFF** BIP errors are not counted (default). 1_B **ON** BIP errors are counted.
8	This mode enables the use of the value 0 as a downstream PLOAM sequence number (SeqNo). This is not used by definition of ITU-T G.987/G.9807/G.989 but nevertheless some OLTs and OLT emulators use the value of 0. If there is an issue with losing of PLOAM messages with the sequence number 0, this option should be enabled. 0_B **OFF** PLOAM downstream messages with a sequence number of 0 are ignored (default). 1_B **ON** PLOAM downstream messages with a sequence number of 0 are handled.
9	This mode modifies the DBRu reporting. This is required to interoperate with specific OLTs. For an empty allocation the value of 0 is sent instead of 0xFFFFFF. 0_B **OFF** Regular DBRu reporting (default). 1_B **ON** Modified DBRu reporting. *Note: For SYS_1.22.0 and later this bit is reserved. The modified DBRu reporting is always used.*
10	This mode modifies the handling of the data requesting by the PON for upstream data. This is required to interoperate with specific OLTs. If activated, a minimum requesting size of 10 kbyte is applied. 0_B **OFF** Regular requesting (default). 1_B **ON** Minimum requesting size.
11 to 31	Reserved

Note: Option bits 3 and 11 to 31 are reserved for future enhancements. They must be set to 0.

2.4.3.4 UCI Configuration – ITU TWDM PON

These TC layer configuration values are only used when the multi-wavelength ITU-T Time and Wavelength Division Multiplex (TWDM) PON operation modes is selected (NG-PON2).

Note: Additional configuration for TWDM is required in /etc/config/optic, see ***Chapter 2.4.3.8****.*

For example:

```
# cat /etc/config/gpon

config gpon 'ploam'
        option nSerial 'MAXL12345678'
        option regID '0x49 0x4e 0x54 0x45 0x4c 0x20 0x46 0x41 0x4c 0x43 0x4f 0x4e
0x20 0x4d 0x4f 0x55 0x4e 0x54 0x41 0x49 0x4e 0x20 0x4c 0x51 0x44 0x45 0xF1 0x50 0xED
0xF3 0x20 0x20 0x20 0x20 0x20 0x20'
        option ploam_timeout_0 '65535'
        option ploam_timeout_1 '10000'
        option ploam_timeout_2   '125'
        option ploam_timeout_3    '25'
        option ploam_timeout_4   '125'
        option ploam_timeout_5   '125'
        option ploam_timeout_6 '10000'
        option ploam_timeout_cpl  '80'
        option ploam_timeout_cpi '300'
        option plev_cap            '0'

config gpon 'ponip'
        option pon_mode 'ngpon2_10G'
        option iop_mask '18'
```

TWDM PON Operation Mode Settings

Table 4 TWDM PON Operation Mode Configuration

Group	Option	Unit	Range	Default	Notes
ploam	nSerial	ASCII	Up to 12 characters	`MAXL12345678`	This is the serial number used by the Physical Layer OAM (PLOAM) protocol. The value is represented as a 12-byte ASCII string.
	regID	–	Up to 36 byte	`0x49 … 0x20`	36-byte Registration ID used by the PLOAM registration protocol for XG-PON, XGS-PON, and NG-PON2. The value is presented as 36 individual bytes in hexadecimal notation to form an ASCII string. In G-PON operation mode, the rightmost (lower) ten bytes of the Registration ID are used as the PLOAM password.
	timeout_0	ms	0 to 65535	`65535`	This timeout value (TOZ) is used to return from PLOAM state O23 in XG-PON, XGS-PON, and NG-PON2 operation mode. This value is not applicable to G-PON operation mode. When set to 0, the timeout is disabled.

Table 4 TWDM PON Operation Mode Configuration (cont'd)

Group	Option	Unit	Range	Default	Notes
ploam	timeout_1	ms	0 to 65535	10000	This value configures the value of TO1. The value is used as ranging timeout (in PLOAM state O4). This value is applicable to all ITU PON operation modes.
	timeout_2	ms	0 to 65535	125	This value configures the value of TO2 (LODS re-initialization timer) in units of 1 ms. Intermittent LODS timeout (in O6 for XG-PON and XGS-PON, and NG-PON2 operation mode) or POPUP timeout (O6 in G-PON operation mode). This value is applicable to all ITU PON operation modes
	timeout_3	ms	0 to 65535	25	This value configures the value of TO3 (LODS protection timer). Intermittent LODS timeout (in O6), if wavelength protection is on (NG-PON2 only).
	timeout_4	ms	0 to 65535	125	This value configures the value of TO4 (downstream tuning timer) for the downstream tuning timeout (in O8, NG-PON2 only).
	timeout_5	ms	0 to 65535	125	This value configures the value of TO5 (upstream tuning timer) for the upstream tuning timeout (in O9, NG-PON2 only).
	timeout_6	ms	0 to 65535	10000	This value configures the value of TO6 for the Lost ONU timeout (triggers if no allocations are received in PLOAM state O5.x). If set to 0, the timeout is disabled.
	timeout_cpl	ms	1 to 65535	80	This value configures the value of TO_CPL for the change power level timeout. A value of 0 is not allowed and causes the initialization to be terminated.
	timeout_cpi	s	0 to 65535	300	This value configures the initial value of the channel partition waiver timer. The value is reconfigurable by the OLT using the OMCI protocol. The value of 0xFFFF indicates infinity (no channel partition waiver is granted).
	plev_cap	bitmap	0 to 127	0	Power level capabilities as defined by ITU-T G.989.3, chapter 11.2.6.8. Power leveling capability is a seven-bit bitmap of the form 0CCC CCCC, whereby a bit in the K-th least significant position indicates that the ONU supports the attenuation level of 3K dB. For example, 00000010 indicates support of 6 dB attenuation level.

Table 4 TWDM PON Operation Mode Configuration (cont'd)

Group	Option	Unit	Range	Default	Notes
ponip	pon_mode	–	–	xgspon	These are the supported TDM PON and active optical Ethernet (AON) operation modes: • gpon • xgpon • xgspon • aon • ngpon2_2G5 • ngpon2_10G
	iop_mask	–	See Table 3.	18	Table 3 shows the available interoperability options for iop_mask. Each of the bits are individually set or cleared to enable or disable the related interoperability option.

2.4.3.5 UCI Configuration – IEEE 1588

These configuration values are only used when the PTP protocol as defined by IEEE 1588 is in use. The information is used by the PTP software daemon (ptp4l). For example:

```
# cat /etc/config/ieee_1588

config ieee_1588 'ptp_sync'
        option delay_mechanism '-E'
        option network_transport '-2'
        option timestamping '-H'
        option cfg_file '/etc/g_8275_1.conf'
        option log_level '6'
        option interface 'eth0_0'

config ieee_1588 'tod2phc'
        option offset '0'
        option time_thr '4'
        option sync_offset '-14'
        option log_level '6'
        option xpectd_mc_name 'PONIP TOD 1'
        option xpectd_sc_name 'xgmac_clk'
        option slave_clock 'ptp2'
        option master_clock 'ptp3'

config ieee_1588 'tod2ktimer'
        option auto_cfg '-a'
        option synch_sys_clk '-r'
        option domain_num '24'
        option offset '0'
        option log_level '6'
        option xpectd_phc_name 'PONIP TOD 1'
        option phc_clk 'ptp3'

# cat /etc/g_8275_1.conf
[global]
```

```
#
# Default Data Set for G.8271.1 PTP Telecom Profile
#
twoStepFlag 1
slaveOnly 0
masterOnly 1
priority1 128
priority2 128
domainNumber 24
clockClass 248
clockAccuracy 0xFE
offsetScaledLogVariance 0xFFFF
free_running 0
freq_est_interval 1
#
# Port Data Set
#
logAnnounceInterval -3
logSyncInterval -4
logMinDelayReqInterval -4
logMinPdelayReqInterval 0
announceReceiptTimeout 3
syncReceiptTimeout 0
delayAsymmetry 0
fault_reset_interval 4
neighborPropDelayThresh 20000000
#
# Run time options
#
assume_two_step 1
logging_level 6
path_trace_enabled 0
follow_up_info 0
hybrid_e2e 0
tx_timestamp_timeout 100
use_syslog 1
verbose 0
summary_interval 0
kernel_leap 1
check_fup_sync 0
#
# Servo Options
#
pi_proportional_const 0.0
pi_integral_const 0.0
pi_proportional_scale 0.0
pi_proportional_exponent -0.3
pi_proportional_norm_max 0.7
pi_integral_scale 0.0
pi_integral_exponent 0.4
pi_integral_norm_max 0.3
```

```
step_threshold 0.0
first_step_threshold 0.00002
max_frequency 900000000
clock_servo pi
sanity_freq_limit 200000000
ntpshm_segment 0
#
# Transport options
#
transportSpecific 0x0
ptp_dst_mac 01:1B:19:00:00:00
p2p_dst_mac 01:80:C2:00:00:0E
udp_ttl 1
udp6_scope 0x0E
uds_address /var/run/ptp4l
#
# Default interface options
#
network_transport L2
delay_mechanism E2E
time_stamping hardware
tsproc_mode filter
delay_filter moving_median
delay_filter_length 10
egressLatency 40
ingressLatency 102
boundary_clock_jbod 0
#
# Clock description
#
productDescription ;;
revisionData ;;
manufacturerIdentity 00:00:00
userDescription ;
timeSource 0xA0
```


2.4.3.6 UCI Configuration – Network

This file provides the network and network interface configurations. The MAC and IP addresses must be individually configured per ONU during production. For example (after boot-up without OLT connection):

cat /etc/config/network

```
config interface 'loopback'
        option device 'lo'
        option proto 'static'
        option ipaddr '127.0.0.1'
        option netmask '255.0.0.0'

config globals 'globals'
        option ula_prefix 'fd11:7d0f:30c7::/48'
```

```
config interface 'lct'
        option ifname 'eth0_0_1_lct'
        option ipaddr '10.91.184.195'
        option netmask '255.255.224.0'
        option proto 'static'
        option auto '0'
        option macaddr 'ac:9a:96:fb:f4:00'
        option gateway '10.91.160.1'

config interface 'iphost1'
        option ifname 'iphost1'
        option proto 'static'
        option auto '0'
        option ipaddr '0.0.0.0'
        option netmask '0.0.0.0'
        option macaddr 'ac:9a:96:fb:f4:05'

config interface 'ip6host1'
        option ifname 'ip6host1'
        option proto 'static'
        option auto '0'

config device 'iphost1_bp'
        option name 'iphost1_bp'

config interface 'iphost2'
        option ifname 'iphost2'
        option proto 'static'
        option auto '0'
        option ipaddr '0.0.0.0'
        option netmask '0.0.0.0'
        option macaddr 'ac:9a:96:fb:f4:06'

config interface 'ip6host2'
        option ifname 'ip6host2'
        option proto 'static'
        option auto '0'

config device 'iphost2_bp'
        option name 'iphost2_bp'
```

2.4.3.7 UCI Configuration – OMCI

These configuration values are only used by the OMCI daemon in case of ITU-T PON applications.

The default part is mainly used to define the MIB initialization files and the network interface used for OMCI message handling. The values for `omci_lpwd` and `omci_loid` must be individualized per ONU during production.

The `iphost` part defines the traffic rate limitation for the IP Host network interface to avoid software overload by DoS attacks.

For example, on a live system:

```
# cat /etc/config/omci

config omci 'default'
        option cur_mib_file '/etc/mibs/current_mib.ini'
        option loid 'loid'
        option lpwd 'lpasswd'
        option omcc_version '0xA3'
        option dbg_lvl '3'
        option lct_disable '0'
        option warmstart '0'
        option mib_data_sync '0'
        option traffic_management '2'
        option enabled '1'
        option omcc_netdev 'gem-omci'
        option mib_file '/etc/mibs/prx300_2U.ini'
        option lct_port '1'

config omci 'iphost'
        option enabled '1'
        option committed '32000 2048'
        option peak '32000 2048'
        option color_aware '0'
        option mode '0'

config device 'uni1'
        option name 'eth0_0'
        option macaddr '00:e0:92:00:01:42'

config device 'uni2'
        option name 'eth0_1'
        option macaddr '00:e0:92:00:01:43'

config omci 'net'
        option enable_cpu_queues '1'

config omci 'pon_adapter'
        list modules 'libponimg'
        list modules 'libponnet'
        list modules 'libpon'
```

```
config device 'veip1'
        option name 'eth0_0'
        option multicast_name 'eth0_0_2'
        option broadcast_name 'eth0_0_3'
```

Table 5 shows the configuration items.

Table 5 OMCI Configuration

UCI Item	Default Value	Usage
config omci 'default'		
option mib_file	/etc/mibs/ponip.ini	Factory default MIB initialization file. Used when the OLT requests a MIB reset. See Section 2.4.3.7.1 for more details.
option cur_mib_file	/etc/mibs/current_mib.ini	Last used MIB status file. Used for OMCI start-up.
option mib_status	–[1)]	This is the path to the file that is used to store the latest MIB configuration status. The file is updated each time a change of the MIB is done by the OLT. This file can be read by other management systems to gain information from the OMCI.
option mib_status_counter	–[1)]	This is the path to the file that is used to store the latest MIB counter information. The file is updated in regular intervals. This file can be read by other management systems to read counter values from the OMCI.
option omci_lpwd	lpasswd	Logical password reported by ONU-G, ASCII string, limited to 12 bytes.
option omci_loid	loid	Logical ONU ID reported by ONU-G, ASCII string, limited to 24 bytes.
option omcc_netdev	gem-omci	Linux Ethernet network interface used to connect the OMCI daemon. OMCI messages are received and transmitted on this interface.
option iop_mask	–[1)]	OMCI interoperability configuration mask. Each bit enables one specific interoperability function. See Table 6. If this option is not present, all OMCI interoperability options are disabled.[2)]
option omcc_version	163 or 0xA3	This defines the OMCC version that is supported. The version 164/0xA4 is also supported by the reference implementation.
option lct_disable	0	This enables the LCT interface by default. If the LCT administrative state can be handled by the OLT, this must be set to 1 (administrative state is locked). If this cannot be set through the OLT and the LCT access is needed, it must be set to 0 (administrative state is unlocked).
option lct_port	1	This defines the OMCI connection of the LCT interface. See Chapter 4.20.7.1 for more detail.

Table 5 OMCI Configuration (cont'd)

UCI Item	Default Value	Usage
config omci 'iphost'		
option enabled	1	These options control the allowed data rate accepted by the IP Host network interface.
option committed	32000 2048	
option peak	32000 2048	
option color_aware	0	
option mode	0	
config device 'uni1'		
option name	eth0_0	This defines the Linux network device name for the local UNI1 interface.
option macaddr	Variable	The MAC address must be defined to be unique.
config device 'uni2'		
option name	eth0_1	This defines the Linux network device name for the local UNI2 interface.
option macaddr	Variable	The MAC address must be defined to be unique.
config omci 'net'		
option enable_cpu_queues	1	Enables QoS for the OMCI network interface.
config omci 'pon_adapter'		
list modules	libponimg	Software modules required for the PON adapter layer.
list modules	libponnet	
list modules	libpon	
config device 'veip1'		
option name	eth0_0	This defines the Linux network device name for the VEIP interface (if present).
option multicast_name	eth0_0_2	This defines the Linux multicast network device name for the VEIP interface (if present).
option broadcast name	eth0_0_3	This defines the Linux broadcast network device name for the VEIP interface (if present).

1) A default value is not part of the 10G PON Chipset System Package delivery.
This configuration can be added according to the needs of the target application.

2) If the U-Boot variable *omci_iop_mask* exists, this value is copied into UCI by the OMCI startup script */etc/init.d/omcid.sh*.

2.4.3.7.1 MIB Initialization File Handling

To change the default values or create a vendor-specific MIB initialization file on the build system, the new file(s) must be located at `ugw_sw/feeds/ugw/pon/gpon-omci-onu/files/etc/mibs/`.

The UCI configuration file is located at
`ugw_sw/feeds/ugw/pon/gpon-omci-onu/files/etc/config/omci`.

The OMCI start-up script is stored at
`ugw_sw/feeds/ugw/pon/gpon-omci-onu/files/etc/init.d/omcid.sh`.

To select a different default MIB initialization file, this is done in the script `ugw_sw/feeds/ugw/pon/gpon-omci-onu/files/etc/uci-defaults/omci-set-mib-file`.
This script selects the default MIB initialization file based on the hardware board type. The reference board types for PRX120, PRX126, and PRX321 are covered by default. It is possible to add the vendor-specific board types here.

The OMCI MIB initialization file can also be selected based on a U-Boot environment variable. When the variable `mib_file` is set to a valid file name, the previous selection is ignored. This is handled by the OMCI start-up script `omcid.sh`.

2.4.3.7.2 OMCI Interoperability Settings

The available interoperability options are shown in **Table 6**. Each of the bits can be individually set or cleared. The options are enabled by setting the respective bit.

Table 6 OMCI Interoperability Settings

IOP Bit	OMCI Interoperability Function
0	Force VLAN-unaware IGMP/MLD behavior. IGMP/MLD packets are services independent of their VLAN properties.
1	Skip Alloc-IDs termination upon T-CONT deactivation. If a T-CONT is deactivated in OMCI, the related Allocation ID is still filled with GEM idle frames, if requested by the OLT.
2	Drop all packets on default DS Extended VLAN rules.
3	Ignore DS Extended VLAN rules priority matching.
4	Convert Traffic Descriptor CIR/PIR values from kbit/s to bytes/s. This must be selected if the OLT uses values given in kbit/s for the CIR/PIR values instead of bytes/s (as defined by OMCI).
5	Force common IP handling - apply the IPv4 Ethertype 0x0800 to the Extended VLAN rule matching for IPv6 packets that contain an Ethertype of 0x86DD.
6	Enable padding of short Extended OMCI packets (less than 48 byte) in upstream direction.
7	Reserved
8	XGS-PON Encryption Key Ring interoperability handling. When the "Encryption Key Ring" is received with a value of 0, it will be handled in the same way as if it was received as 1.
9 to 31	The option bits are reserved for future enhancements. They must be set to 0.

2.4.3.8 UCI Configuration – Optical Interface

These configuration values are used to setup the optical interface. The setup and hold time values depend on the external optical components and their placement on the PCB and must be evaluated individually for each ONU implementation. The configurable items are described in **Table 7**.

For example:

```
# cat /etc/config/optic
config 'optic' 'common'
        option 'bias_hold_time'          '202545'
        option 'bias_setup_time'             '0'
        option 'serdes_hold_time'         '35365'
        option 'serdes_setup_time'           '0'
        option 'laser_hold_time'             '0'
        option 'laser_setup_time'            '0'
        option 'burst_idle_pattern' '0x0000FFFF'
        option 'burst_en_mode'               '0'
        option 'tx_en_mode'                  '0'
        option 'tx_pup_mode'                 '0'
        option 'loop_timing_mode'            '3'
        option 'loop_timing_power_save'      '0'
        option 'sd_polarity'                 '1'
        option 'dg_dis'                      '0'
        option 'rogue_lead_time'             '0'
        option 'rogue_lag_time'              '0'
        option 'rogue_auto_en'               '2'
        option 'opt_tx_sd_polarity'          '0'
        option 'gpio21_mode'                 '0'
        option 'gpio24_mode'                 '0'
        option 'gpio25_mode'                 '0'
        option 'tx_power_scale'              '0'
        option 'transceiver_name'            '<name>'
        option 'version'                     '<version>'

config 'optic' 'threshold'
        option 'lower_receive_optical_threshold'  '-60'
        option 'upper_receive_optical_threshold'  '-14'
        option 'lower_transmit_power_threshold'     '6'
        option 'upper_transmit_power_threshold'    '20'
        option 'bias_threshold'                 '60000'

config optic 'eeprom'
        option dmi '/sys/bus/i2c/devices/0-0051/eeprom'
        option serial_id '/sys/bus/i2c/devices/0-0050/eeprom'

config optic 'sfp_eeprom'
        option dmi '/sys/class/pon_mbox/pon_mbox0/device/eeprom51'
        option serial_id '/sys/class/pon_mbox/pon_mbox0/device/eeprom50'

config 'optic' 'twdm'
        option 'channel_count'              '4'
        option 'channel_mask'            '0x0F'
```

```
option 'config_method'              '3'
option 'rx_wl_switch_time'        '8000'
option 'tx_wl_switch_time'        '8000'
option 'wl_switch_delay'         '24000'
option 'free_spectral_range'      '3000'
option 'wl_ch_spacing'             '100'
option 'spectral_excursion'         '20'
option 'optical_link_type'           'A'
option 'tx_power_min'                '4'
option 'tx_power_max'                '9'

option 'channel_partition_index'     '0'
option 'dwlch_id'                    '0'

option 'cal_status_record' '0x3333000000000000'

option tuning                        '0'
option tuning_gran                   '0'
option fine_tune_time                '0'
```

Table 7 Optical Configuration Values

Group	Option	Unit	Range	Default	Notes
Common Settings					
common	bias_hold_time *Note: This is the time between the end of the data burst and the disabling of the laser (the falling edge of OPT_TXEN).*	3215 ps	0 to 65535, resulting in integer multiples of the clock cycle time[1)]	Application specific	This value depends on the laser component type in use and the PCB layout.
	bias_setup_time *Note: This is the time at which the laser is enabled (through OPT_TXEN) related to the begin of the data burst. A value of 0 is the earliest possible enable.*	3215 ps	0 to 65535 resulting in integer multiples of the clock cycle time[1)]	0 (minimum value)	
	serdes_hold_time *Note: To save power, this forces tx_disable to be earlier at the end of the burst.*	ps	0 to 1023 given in integer multiples of the clock cycle time[1)]	3288945 (maximum value)	
	serdes_setup_time *Note: To save power, this forces tx_disable to be delayed at the start of the burst.*	ps	0 to 4095 given in integer multiples of the clock cycle time[1)]	0	
	laser_hold_time	ps	0 to 1023 given in integer multiples of the clock cycle time[1)]	12860	
	laser_setup_time	ps	0 to 4095 given in integer multiples of the clock cycle time[1)]	0	
	burst_en_mode	–	0_H **SHORT** Directly controlled for short burst start times (default mode) 1_H **LONG** Indirectly controlled to allow larger burst start times.	0	

Table 7 Optical Configuration Values (cont'd)

Group	Option	Unit	Range	Default	Notes
common	burst_idle_pattern	–	32-bit binary bit pattern[2)]	0x0000FFFF	This pattern is sent to the laser driver between bursts to support AC coupling.
	tx_en_mode	–	0_H **AUTO** Automatic level detection 1_H **HIGH** Active high output level 3_H **LOW** Active low output level	0	This selects the polarity of the OPT_TXEN output signal that is driven to the PMD. The automatic mode is recommended.
	tx_pup_mode	–	1_H **LOW** Active low output level 3_H **HIGH** Active high output level	0	This selects the polarity of the OPT_TX_PUP output signal that is optionally driven to the PMD.
	loop_timing_mode	–	0_H **RES** Reserved, do not use 1_H **RES** Reserved, do not use 2_H **RES** Reserved, do not use 3_H **FW** The firmware handles the loop timing control	3	Always use FW-controlled loop timing.

Table 7 Optical Configuration Values (cont'd)

Group	Option	Unit	Range	Default	Notes
common	loop_timing_power_save	–	0_H **DIS** Disabled 1_H **EN** Enabled	0	Enable power saving by reducing the internal VCO frequency from 10 GHz to 5 GHz.
	dg_dis	–	0_H **EN** DG is enabled 1_H **DIS** DG is disabled	0	Optionally disable sending Dying Gasp information to the OLT.
	rogue_lead_time	ns	0 to 3000	0[3)]	3.2 ns granularity
	rogue_lag_time	ns	0 to 3000	0[3)]	3.2 ns granularity
	rogue_auto_en	–	0_H **DIS** Automatic detection is disabled 1_H **EN** Automatic internal detection is enabled 2_H **EXT** Automatic external detection is enabled	0[3)]	External Rogue ONU detection is the recommended operation mode when the PMD supports this. Automatic internal mode requires an external connection of the signal OPT_TX_SD to the PMD.
	opt_tx_sd_polarity	–	0_H **LOS** Loss of signal function 1_H **SD** Signal detect function	0	This option defines the polarity of the input signal OPT_TX_SD. This transmit signal detect information is provided by the PMD.
	sd_polarity	–	0_H **DIR** Direct signal detect function 1_H **INV** Inverted signal detect function	0	This option defines the polarity of the input signal OPT_RX_LOS. The receive signal detect indication is provided by the PMD.
	gpio21_mode	–	0_H **DEFAULT** The signal is driven by the PON IP hardware	0	This option controls the usage of OPT_TX_DIS_SFP. Only the default operation is supported. This signal is available only on PRX126.

Table 7 Optical Configuration Values (cont'd)

Group	Option	Unit	Range	Default	Notes
common	gpio24_mode	–	0_H **HW** The signal is driven by the PON IP hardware 1_H **FW** The signal is overridden by the PON firmware	0	This option controls the usage of OPT_TX_PUP. Changes are only required for PRX126 applications.
	gpio25_mode	–	0_H **DEFAULT** LOS is forwarded from the PMD 1_H **LOW** Fixed low voltage level 2_H **HIGH** Fixed high voltage level 3_H **HIMP** High impedance	0	Changes are only required for SFP usage in PTP timing test equipment.
	tx_power_scale	–	0_H **DEF** Default factor 0.1 µW/bit 1_H **BY2** Factor of 0.2 µW/bit	0	This setting is used to adapt to PMD devices that report the transmit power in a scale different from 0.1 µW as defined in [67]. A reporting unit of 0.2 µW can optionally be supported.
	transceiver_name	–	ASCII string	–	See **Optical Configuration Database**
	version	–	ASCII string	–	

Table 7 Optical Configuration Values (cont'd)

Group	Option	Unit	Range	Default	Notes
Default Optical Thresholds					
These values trigger optical OMCI alarms when no other values are specified by the OLT through the OMCI protocol (see **ANI-G**).					
threshold	lower_receive_optical_threshold	0.5 dBm	-127 dBm to 0 dBm	-60 (-30 dB)	A value of 0xFF selects the hard-coded internal threshold.
	upper_receive_optical_threshold	0.5 dBm	-127 dBm to 0 dBm	-14 (-7 dB)	
	lower_transmit_power_threshold	0.5 dBm	-127 dBm to 0 dBm	6 (3 dB)	A value of 0x81 selects the hard-coded internal threshold.
	upper_transmit_power_threshold	0.5 dBm	-127 dBm to 0 dBm	20 (10 dB)	
	bias_threshold	2 µA	0 to 200 mA	60000 (120 mA)	This value depends on the laser component type in use.
PMD Path Information					
These are the logical paths to access the PMD device over the I²C bus. The UCI information is automatically generated by a script upon the first system boot.					
eeprom[4)]	dmi	–	–	/sys/bus/i2c/devices/0-0051/eeprom	Path to access the dynamic DDMI information of the PMD.
	serial_id	–	–	/sys/bus/i2c/devices/0-0050/eeprom	Path to access the static information of the PMD.
sfp_eeprom[5)]	dmi	–	–	/sys/class/pon_mbox/pon_mbox0/device/eeprom51	Path to access the dynamic DDMI information of the PMD.
	serial_id	–	–	/sys/class/pon_mbox/pon_mbox0/device/eeprom50	Path to access the static information of the PMD.

Table 7 Optical Configuration Values (cont'd)

Group	Option	Unit	Range	Default	Notes
Default TWDM Settings					
These values are used to configure the TWDM parameters used for NG-PON2 systems[6].					
twdm	channel_count	–	1 to 8	4	Number of supported wavelength channel pairs.
	channel_mask	–	8 bits	0x0F	Bit mask of supported wavelength channel index pairs. Bit position 0 refers to wavelength channel index 0.
	config_method	–	0, 1 to 3[7]	0	The applicable value depends on the optical PMD device. The special value 0 disables the wavelength selection.
	rx_wl_switch_time	125 µs	6400 or larger	8000 (1 second)	This is the worst case time needed to switch to a different receive wavelength.
	tx_wl_switch_time	125 µs	6400 or larger	8000 (1 second)	This is the worst case time needed to switch to a different transmit wavelength.
	wl_switch_delay[8]	125 µs[9]	0 or larger	24000 (3 seconds)	This is slowing down the wavelength search if no signal applied to save power.
	free_spectral_range	0.1 GHz	0 to 65535	3000	If a cyclic wavelength multiplexer is used in the upstream, this is an unsigned integer indicating the value in units of 0.1 GHz. If not, use 0.
	wl_ch_spacing	1 GHz	0 to 255	100	Channel spacing is a system parameter characterizing the grid to which the system is tuned.
	spectral_excursion	1 GHz	0 to 255	20	Maximum spectral excursion

Table 7 Optical Configuration Values (cont'd)

Group	Option	Unit	Range	Default	Notes
twdm	optical_link_type	–	Unspecified: 0 A only: 2 B only: 1 A and B: 3	A	Optical link type support
	tx_power_min	dB	2 to 4	4	Minimum optical transmit power
	tx_power_max	dB	7 to 9	9	Maximum optical transmit power
	channel_partition_index	–	0 to 15	0	Channel Partition Index (CPI) as defined in ITU-T G.989.3, chapter 6.1.5.9.
	dwlch_id	–	0 to 3	0	Downstream wavelength channel ID that was used last.
	cal_status_record	–	0x0000 0000 0000 0000 to 0x3333 3333 3333 3333	0x3333 0000 0000 0000	The optical calibration record specifies the quality and usability of the upstream wavelengths
	tuning	–	0 or 1	0	Upstream fine-tuning capability. Set to 0 if not supported.
	tuning_gran	1 GHz	0 to 255	0	Upstream fine-tuning granularity. Set to 0 if not supported.
	fine_tune_time	125 µs	0 to 255	0	Upstream fine-tuning time per step. Set to 0 if not supported.

1) ITU PON modes: Operation is based on 311.04 MHz, 3215 ps per clock cycle. Only multiples of the clock cycle time are configurable.
2) The hardware supports a pattern length of up to 256 bits, software support will be added in the future.
3) The timing values and the polarity must be calibrated to a value matching the used optical transceiver type, before automatic rogue ONU detection can be enabled.
4) This is used up to 10G PON Chipset System Package 1.16.x and kept for backward compatibility with future versions.
5) This is used starting from 10G PON Chipset System Package 1.17.0.
6) These values depend on the optical interface and PMD type.
7) Additional methods may be supported in future.
8) This function is available from SYS_1.22.0 onwards.
9) The value is rounded to an effective delay time granularity of 1 ms.

Optical Configuration Database

For known optical interfaces, there is a database provided with the software.

The database is a simple directory, holding a single file for each known optical interface type. The file names are constructed as: `<vendor_name>-<part_number>` (spaces are removed).

For example:

vendor_name = "VENDOR XY LTD."

part_number = "123456-ABC"

results in file name "vendorxyltd-123456-abc"

```
:/rom/etc/optic-db# ll
drwxr-xr-x    2 root     root      331 Jul  4 20:54 ./
drwxr-xr-x   19 root     root      829 Jul  4 20:04 ../
-rw-r--r--    1 root     root      431 Jul  4 20:04 vendorxyltd-123456-abc
```

Upon the first start of the software (or after a factory reset), the optical interface (or PMD) is accessed through the I^2C bus and the type data is read.

To identify the transceiver or PMD type the values of "Vendor Name" (at 0xA0, bytes 20 to 35) and "Vendor Part Number" (at 0xA0, bytes 40 to 55) are read.

The detected transceiver type (transceiver_name) is stored together with the software binary version (version) in the UCI configuration file `/etc/config/optic`. Upon each subsequent system start-up, the stored software version and transceiver type are checked against the current software version and transceiver type. If one of them differs, the transceiver detection is performed again to apply possible configuration updates that can happen during a software image update or a change of the optical transceiver type[1)].

If no matching entry is found, the existing default values are used and not overwritten.

This functionality is covered by the script located at `/lib/pon/pondb.sh`.

The recommended way to adapt to other optical interfaces is to add related configuration data files to the database folder.

If a group of optical interfaces uses an identical configuration, logical links can be used to a single configuration file. For example:

```
picadvanced-pan2ssxssa3c
picadvanced-pan2ssxssa3i -> picadvanced-pan2ssxssa3c
picadvanced-pan2suxssa3 -> picadvanced-pan2ssxssa3c
picadvanced-pan2suxssa3c -> picadvanced-pan2ssxssa3c
picadvanced-pan2xuxssa -> picadvanced-pan2ssxssa3c
```


Optical Enhancements for TWDM PON

There are optical transceivers that need individual configuration values depending on the selected upstream data rate. To support this, a dedicated section is provided in the configuration file. Based on the selected upstream data rate, the related configuration values are copied into the target configuration file.

The section key words are "ngpon2_10G" and "ngpon2_2G5" as defined for the "pon_mode" configuration in `/etc/config/gpon`.

An example is given below.

```
config optic 'common'
        option bias_hold_time '202545'
        option bias_setup_time '0'
        option laser_hold_time '0'
        option laser_setup_time '0'
        option serdes_hold_time '3288945'
        option serdes_setup_time '0'
        option burst_en_mode '0'
```

1) Hot-plugging of an optical transceiver is not supported.

```
config optic 'ngpon2_10G'
        option bias_hold_time 'xxx'
        option bias_setup_time '0'
        option laser_hold_time '0'
        option laser_setup_time 'yyy'
        option burst_en_mode '1'


config optic 'ngpon2_2G5'
        option bias_hold_time 'aaa'
        option bias_setup_time '0'
        option laser_hold_time '0'
        option laser_setup_time 'bbb'
        option burst_en_mode '1'
```

Optical Database Reference

These optical interface types are supported by the 10G PON Chipset System Package 1.17.0 and later.

```
acalbfi-fhm-5c20si-2757
acalbfi-fhm5820la-34avm2
default
eoptolinkinc-eols-gu-25-d
eoptolinkinc-eols-gu-25-dj
eoptolinkinc-eols-gu-25-dj-urx851-eva
eoptolinkinc-eols-gu-25-dj-urx851-ref
hisense-ltf7225-bc-urx851-ref
intel-m2180
intel-m2181
ligentphotonics-ltw2601cbc
lightroninc-0013c5-1wekrxxx8a
macom-m2180
macom-m2181
mentechopto-mpos22-ldcc-co-urx851-ref
mentechopto-msog22-ldcc-acco-urx851-ref
municom-munsfpgpononubda
municom-munsfpgpononubda-urx851-eva
municom-munsfpgpononubda-urx851-ref
mxlprx126-gn28l96
sourcephotonics-spps27xer3cdfd
superxonltd-sogx2699-psga
superxonltd-sogx2699-psga-urx851-eva
superxonltd-sogx2699-psga-urx851-ref
wtd-rtxm166-401
wtd-rtxm166-501
wtd-rtxm167-431
```

Optical PMD Detection without EEPROM at 0x50

If the implementation uses a PMD that does not provide EEPROM access to address 0x50, then the related contents must be stored in a binary file. This file must be referenced in `/etc/config/optic`, for example as

```
option serial_id '/tmp/my_eeprom_50.bin'
```

Create this binary file from a hexadecimal file, for example by using the `xxd` tool:

```
xxd -r -p my_eeprom_50.txt my_eeprom_50.bin
```

It is possible to create a reference hexadecimal file by copying and modifying the contents of the dump of an existing transceiver with EEPROM. Here is an example:

```
# hexdump -Cv /sys/class/pon_mbox/pon_mbox0/device/eeprom50
00000000  03 04 01 00 00 00 00 00  00 00 00 03 63 00 14 c8  |............c...|
00000010  00 00 00 00 53 55 50 45  52 58 4f 4e 20 4c 54 44  |....SUPERXON LTD|
00000020  2e 20 20 20 00 00 00 00  53 4f 47 58 32 36 39 39  |........SOGX2699|
00000030  2d 50 53 47 41 20 20 20  31 30 20 20 04 f6 00 ce  |-PSGA   10  ....|
00000040  00 1c 00 00 39 37 39 36  31 31 38 33 32 30 30 32  |....979611832002|
00000050  30 35 20 20 31 38 30 38  31 37 20 20 68 f0 02 04  |05  180817  h...|
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000f0  34 41 34 31 37 34 36 30  30 30 39 30 30 20 20 20  |4A41746000900   |
```


PON Database Script

The PON database script is executed upon an initial boot process. It is located at `/lib/pon/pondb.sh`.

```
#!/bin/sh
# shellcheck shell=dash

OPTIC_DB_LOCATION="/etc/optic-db"
SERDES_DB_LOCATION="/etc/serdes-db"

source /lib/pon.sh

log_console() {
    echo "$@" > /dev/console
}

to_lower() {
    #tr on busybox doesn't support classes like [[:upper:]],
    #so this is done by awk
    awk '{print tolower($0)}'
}

normalize() {
    #The sed will throw out characters not suitable for filenames
    #(including whitespace)
```

```
    to_lower | sed -r 's/[^A-Za-z0-9_-]//g'
}

normalize_revision() {
    #same as normalize but will keep the '.' (dot)
    sed -r 's/[^A-Za-z0-9._-]//g'
}

prepend() {
    local prefix="$1"
    shift || return
    sed "s,^,$prefix,"
}

append() {
    local suffix="$1"
    shift || return
    sed "s,$,$suffix,"
}

image_version_get() {
    local ugw_ver

    # prefer the PON version, but if it is not there use the URDK version
    if [ -f /etc/pon.ver ]; then
        image_version=$(cat /etc/pon.ver)
    elif [ -f /etc/version ]; then
        image_version=$(cat /etc/version)
    else
        image_version=""
    fi
}

read_eeprom() {
    local skip="$1"
    shift || return
    local count="$1"
    shift || return

    dd bs=1 if="$EEPROM_PATH" skip="$skip" count="$count" 2> /dev/null
}

transceiver_names_get() {
    local vendor_name="$(read_eeprom 20 16 | normalize)"
    local vendor_oui="$(read_eeprom 37 3 | hexdump -ve '1/1 "%.2x"')"
    local part_number="$(read_eeprom 40 16 | normalize)"
    local revision="$(read_eeprom 56 4 | normalize_revision)"

    #Examples:
    #wtd-001cad-rtsm166-501-1.0
    #wtd-rtxm166-501
```

```
    if [ -n "$vendor_name" ] && [ -n "$part_number" ]; then
        if [ "$vendor_oui" != "000000" ] && [ -n "$revision" ]; then
            echo "$vendor_name-$vendor_oui-$part_number-$revision"
        fi
        if [ -n "$revision" ]; then
            echo "$vendor_name-$part_number-$revision"
        fi
        if [ "$vendor_oui" != "000000"  ]; then
            echo "$vendor_name-$vendor_oui-$part_number"
        fi
        echo "$vendor_name-$part_number"
    fi
}

optic_files_get() {
    local board="$(pon_board_name)"

    if [ -n "$board" ]; then
        # prepend directoy name and append board name
        #Example: /etc/optic-db/superxonltd-sogx2698-psga-urx851-eva
        transceiver_names_get | prepend "$OPTIC_DB_LOCATION/" | append "-$board"
    fi
    #We prepend directory name
    #Example: /etc/optic-db/wtd-001cad-rtsm166-501-1.0
    transceiver_names_get | prepend "$OPTIC_DB_LOCATION/"

    echo "$OPTIC_DB_LOCATION/default"
}

serdes_files_get() {
    local board="$(pon_board_name)"

    if [ -n "$board" ]; then
        #We prepend directory name
        #Example: /etc/serdes-db/prx126-sfp-eva-pon-001cad-rtsm166-501-1.0.conf
      transceiver_names_get | prepend "$SERDES_DB_LOCATION/$board-" | append ".conf"
        echo "$SERDES_DB_LOCATION/$board.conf"
    fi
    echo "$SERDES_DB_LOCATION/default.conf"
}

config_apply() {
    local config_file=""
    local config="$1"
    shift || return

    for filename in "$@"; do
        log_console "[${config}-db] Looking for '$filename' configuration"
        if [ -f "$filename" ]; then
            config_file="$filename"
```

```
            break
        fi
    done

    if [ -n "$config_file" ]; then
        uci -m import "$config" < "$config_file" && uci commit "$config" &&
            log_console "[${config}-db] Applied '$config_file' configuration"
    else
        log_console "[${config}-db] Using default configuration"
    fi
}

EEPROM_PATH="$(uci get optic.eeprom.serial_id)"
[ -z "$EEPROM_PATH" ] && exit 1

image_version_get
# get first name (most detailed) as reference
transceiver_name=$(transceiver_names_get | head -n1)

optic_version=$(uci -q get optic.common.version)
optic_transceiver=$(uci -q get optic.common.transceiver_name)
[ "$optic_version" != "$image_version" ] && optic_change=1
[ "$optic_transceiver" != "$transceiver_name" ] && optic_change=1

if [ $optic_change ]; then
    config_apply optic $(optic_files_get)
    uci set optic.common.version="$image_version"
    uci set optic.common.transceiver_name="$transceiver_name"
    uci commit optic
fi

# we don't have a serdes config for specific transceivers yet.
# In case we will get this, the handling here needs to be extended like above.
serdes_version=$(uci -q get serdes.generic.version)
[ "$serdes_version" != "$image_version" ] && serdes_change=1

if [ $serdes_change ]; then
    config_apply serdes $(serdes_files_get)
    uci set serdes.generic.version="$image_version"
    uci commit serdes
fi
```


Transceiver Types for Multiple PON Modes

Some transceiver types are used for multiple PON operation modes, for example for XG-PON and XGS-PON.

This requires different settings of some timing parameters which are not automatically handled by the transceiver type identification.

In this case, a vendor specific software enhancement is required to enhance the configuration files by operation mode specific settings.

Configuration for Active Optical Ethernet (AON) Operation Mode

For the AON mode, the configuration in `/etc/config/optic` is replaced by:

```
config optic 'common'
        option aon_mode '1' # 0: PON operation mode (default),
                              1: Active Optical Ethernet operation mode
        option aon_pol  '0' # 0: OPT_TX_ENABLE is active high (default),
                              1: OPT_TX_ENABLE is active low
```

Note: To switch the operation mode between one of the PON modes and AON mode, the device must be re-booted, either with a different device tree (if selectable in U-Boot) or a different software image. The 10G PON Chipset System Package creates a dedicated AON ("eth") image.

2.4.3.9 UCI Configuration – SerDes

See **Section 2.4.4.4** for more details.

2.4.3.10 UCI Configuration – SFP

These configuration values are only used when the ONU is implemented as an SFP based on PRX126. It defines the virtual EEPROM addresses and I^2C devices used for DDMI support. There is a dedicated I^2C slave device emulation software and firmware provided. This is available only on PRX126-based ONU systems.

For example:

```
# cat /etc/config/sfp_eeprom
config 'sfp_eeprom' 'default'

config 'sfp_eeprom'             'factory_bridge'
        option 'enable'         '0'
        option 'addr_eeprom_0' '0x4E'
        option 'addr_eeprom_1' '0x4F'
        option 'pmd_0' '/sys/bus/i2c/devices/0-004e/eeprom'
        option 'pmd_1' '/sys/bus/i2c/devices/0-004f/eeprom'
```


2.4.3.11 UCI Configuration – System

This file holds general system configurations.

For example:

```
# cat /etc/config/system

config system
        option timezone     'UTC'
        option ttylogin     '0'
        option log_size     '64'
        option urandom_seed '0'
        option hostname     'prx321-sfu-qspi-pon'
```

2.4.3.12 UCI Configuration – ToD Timing

These configuration values are only used when the ONU supports ToD (Time of Day) serial output. It defines the timing of the 1PPS signal and the serial data output.

Individual configurations are given for specific operation modes.

- `gpon`: This covers the ITU-T PON mode according to G.984.
- `xgspon`: This covers the ITU-T PON modes according to G.987 and G.9807.
- `ngpon2`: This covers ITU-T PON mode according to G.989.

For example:

```
# cat /etc/config/tod
config 'tod' 'common'
        option 'pps_scale' '0'           # < 512
        option 'pps_width' '5000'        # < 8191, unit = 100 us
        option 'pps_delay' '0'           # < unit = 100 us
        option 'pps_polarity' '0'        # 0: rising edge, 1: falling edge
        option 'us_time_correction' '0' # < 1024, unit = internal clock cycles
        option 'ds_time_correction' '0' # < 1024, unit = internal clock cycles

        option 'asymm_corr_gpon'   '0.500065'
        option 'asymm_corr_xgspon' '0.500134'
        option 'asymm_corr_ngpon2' '0.5000727'
        option 'offset_pico_seconds_2g5' '-18119000'
        option 'offset_pico_seconds_10g' '-17241000'
```


Table 8 Time of Day Configuration Values

Group	Option	Unit	Range	Default	Notes
Common Settings					
common	pps_scale		0 … 511	0	
	pps_width	100 µs	0 … 8090	5000	1PPS pulse width, up to ~0.8 s
	pps_delay	100 µs	0 … 10000 - pps_width	0	1PPS pulse delay
	pps_polarity		0/1	0	1PPS pulse polarity
	us_time_correction	clocks	0 …1023	0	
	ds_time_correction	clocks	0 …1023	0	
	asymm_corr_gpon	–		0.500065	Fiber asymmetry factor for G-PON operation mode
	asymm_corr_xgspon	–		0.500134	Fiber asymmetry factor for XG-PON operation mode
	asymm_corr_ngpon2	–		0.5000727	Fiber asymmetry factor for NG-PON2 operation mode
	offset_pico_seconds_2g5	ps		-18119000	Internal delay compensation for G-PON operation mode
	offset_poci_seconds_10g	ps		-17241000	Internal delay compensation for XG-PON, XGS-PON, and NG-PON operation mode

2.4.3.13 UCI Configuration – ToD Daemon

There is a dedicated configuration file for the ToD daemon to configure the PTP device which must be used.

For example:

```
# cat /etc/config/todd
config todd 'daemon'
        option ptp_device '/dev/ptp4'
        option uart '/dev/ttyLTQ1'
        option uart_baudrate '115200'
        option output_mode 'GPZDA'
```

2.4.4 Internal SerDes Configuration

Three SerDes hardware modules are present in the device and are used for serial to parallel data conversion at the LAN interfaces and at the WAN interface. To accommodate different external circuitry conditions, three configuration parameters are adapted per SerDes instance: instances #0 and #1 are used on the LAN side, and instance #2 is used on the WAN side. These must be calibrated once per ONU design prior to production start, based on a hardware prototype.

The correct values must also be provided for the PON IP on the WAN side but are independent of the PCB design. Use the default values delivered with the 10G PON Chipset System Package.

To make the data interfaces available even during the boot phase, the parameters are compiled into the U-Boot code as well as kept in the Linux device tree (application-specific .dts files, see **Section 2.4.2**).

The SerDes configuration is stored at several places:

- U-Boot source code
  - Default settings used to configure the SerDes instances and enable Ethernet access from U-Boot.
- U-Boot environment variables
  - Optional, to modify individual settings and override the hard-coded default settings.
    This is used to configure the SerDes instances manually to optimize the signals.
    It is possible to take the optimized setting into the U-Boot code afterwards.
- Device tree (DTS)
  - These settings are used while the hardware driver is loaded.
  - The settings apply to the LAN-side SerDes instances and to the WAN-side SerDes in case of electrical or active optical Ethernet mode (not used for PON WAN mode).
  - Individual settings are provided by application-specific device tree files.
- UCI configuration
  - These are the settings required to configure the WAN-side SerDes in PON operation mode.

2.4.4.1 SerDes Settings in the U-Boot Source Code

These values are hard coded in the source code for 1 Gbit/s interface speed.

```
afe_en = 0,
dfe_en = 0,
cont_adapt0 = 0,
tx_eq_main = 0x28,
tx_eq_pre = 0,
tx_eq_post = 0,
tx_eq_ovrride = 1,
los_thr = 0,
vboost_lvl = 0x5,
tx_iboost[0] = 0xF,
```

These values are hard coded in the source code for 10 Gbit/s interface speed.

```
afe_en = 1,
dfe_en = 1,
cont_adapt0 = 1,
tx_eq_main = 0x1E,
tx_eq_pre = 0x14,
tx_eq_post = 0x14,
tx_eq_ovrride = 1,
los_thr = 4,
vboost_lvl = 0x5,
tx_iboost[0] = 0xF,
```

2.4.4.2 SerDes Settings in the U-Boot Environment

For lab usage and to ease the calibration process, the U-Boot environment variables as listed below are created to overwrite the compile-time values in the U-Boot source code and are modifiable manually during the calibration process. After the newly calibrated values have been compiled into a new U-Boot image, the environment variables are no longer required and must be deleted from the default U-Boot environment.

- `serdes<0|1|2>_tx_pre`
- `serdes<0|1|2>_tx_post`
- `serdes<0|1|2>_tx_main`
- `serdes<0|1|2>_tx_vboost_en`
- `serdes<0|1|2>_tx_vboost_lvl`
- `serdes<0|1|2>_tx_iboost_lvl`
- `serdes<0|1|2>_rx_eq_ctle_pole`
- `serdes<0|1|2>_rx_eq_ctle_boost`
- `serdes<0|1|2>_rx_los_threshold`

Table 9 SerDes Parameter Value Ranges

Parameter	Minimum Value	Maximum Value	Usage
serdes<0|1|2>_tx_pre	0	63	
serdes<|1|2|3>_tx_post	0	63	
serdes<|1|2|3>_tx_main	0	63	
serdes<|1|2|3>_tx_vboost_en	0	1	
serdes<|1|2|3>_tx_vboost_lvl	0	7	
serdes<|1|2|3>_tx_iboost_lvl	0	15	

2.4.4.3 SerDes Settings in the Device Tree

Several configuration settings depend on the PCB application and are stored in the device tree file (prx300.dtsi). These values must be adapted according to the individual PCP layout. See **Table 10** for details.

Table 10 SerDes Device Tree Settings

Parameter	Value Range	Description
xpcs-mode	• 0: 10G_KR • 1: 10G_XAUI • 2: 1G_XAUI • 3: 2P5G_GMII	Interface mode selection
xpcs-conn	• 0: SFP • 1: PHY	Connection type selection
xpcs-trans-line	• 1: Long • 2: Short	Interface line length. Select short when the attenuation is below 2 dB (typically below about 5 cm of PCB trace).
power-save	• 0: Normal • 1: Power save	Power save mode selection
serdes_1g_cfg	0 to 63	Three numerical values define the pre, post, and main equalizer configuration to adjust the signal form.
serdes_10g_cfg		

Example: LAN-side SerDes – SFP ONU Application

In `prx126-sfp-pon.dts` select the operation mode of the interface:

```
&lan_xpcs1 {
    status = "okay";

    /* 0 - 10G_KR_MODE,
       1 - 10G_XAUI_MODE,
       2 - 1G_XAUI_MODE,
       3 - 2P5G_GMII_MODE
       4 - SUSXGMII
    */
    xpcs-mode = <0>;
};
```

In `prx126-sfp.dtsi` define the SerDes equalization parameters. For each of the selectable modes there is a set of parameters that can be configured. The first parameter is used for `xpcs-mode = <0>`, the last one is used for `xpcs-mode = <4>`.

```
&lan_xpcs1 {
        status = "disabled";
        xpcs-conntype = "SFP"; /* SFP or PHY */
        link_poll_interval = <3>;
        xpcs_mode_cfg_seq = <3 0 2 1>;
        /* xpcs RX EQ Settings */
        xpcs-afe_en = <0x0>, <0x0>, <0x0>, <0x0>, <0x0>;
        xpcs-dfe_en = <0x1>, <0x0>, <0x1>, <0x1>, <0x0>;
        xpcs-cont_adapt_en = <0x1>, <0x1>, <0x1>, <0x1>, <0x0>;
        xpcs-rx_attn_lvl   = <0x7>, <0x7>, <0x7>, <0x7>, <0x7>;
        xpcs-rx_ctle_boost = <0x0>, <0x0>, <0x0>, <0x0>, <0x0>;
        xpcs-rx_ctle_pole  = <0x6>, <0x6>, <0x6>, <0x6>, <0x6>;
        xpcs-rx_vga2_gain  = <0x5>, <0x5>, <0x5>, <0x5>, <0x5>;
        xpcs-rx_vga1_gain  = <0x5>, <0x5>, <0x5>, <0x5>, <0x5>;
        /* xpcs TX EQ Settings */
        xpcs-tx_eq_pre     = <0x0>,  <0x0>,  <0x0>, <0x0>, <0x0>;
        xpcs-tx_eq_post    = <0x0>,  <0x0>,  <0x1>, <0x1>, <0x0>;
        xpcs-tx_eq_main    = <0x28>, <0x28>, <0xC>, <0xC>, <0x28>;
        xpcs-tx_iboost_lvl = <0xF>,  <0xF>,  <0xF>, <0xF>, <0xF>;
        xpcs-tx_vboost_lvl = <0x5>,  <0x5>,  <0x5>, <0x5>, <0x7>;
        xpcs-tx_vboost_en  = <0x1>,  <0x1>,  <0x1>, <0x1>, <0x1>;
};
```


The values given above are examples only and need to be calibrated for the target application. The calibration procedure is described in **[14]**.

2.4.4.4 SerDes Settings in UCI

These configuration values are used to setup the internal SerDes modules connected to the WAN interface. The values depend on the external components and their placement on the PCB and must be evaluated individually for each ONU implementation. See **Section 14.9.2** for the optical interface measurement options.

Table 11 describes the configurable items.

Table 12 and **Table 13** list the values for this example:

```
# cat /etc/config/serdes
```

```
config serdes 'generic'
        option tx_eq_pre         '<value>'
        option tx_eq_post        '<value>'
        option vboost_en         '<value>'
        option vboost_lvl        '<value>'
        option iboost_lvl        '<value>'
        option tx_eq_pre         '<value>'
        option tx_eq_main        '<value>'
        option tx_eq_post        '<value>'
        option rx_slos_thr       '<value>'
        option rx_adapt_cont     '<value>'
        option rx_adapt_afe_en   '<value>'
        option rx_adapt_dfe_en   '<value>'
        option rx_eq_attn_lvl    '<value>'
        option rx_eq_ctle_pole   '<value>'
        option rx_eq_ctle_boost '<value>'
        option rx_eq_vga1_gain   '<value>'
        option rx_eq_vga2_gain   '<value>'
        option rx_eq_dfe_tap1    '<value>'
```

Table 11 PON SerDes Configuration Values

Group	Option	Unit	Range	Notes
Transmit Path				
generic	vboost_en	–	0 to 1	Enables the output voltage boost mode.
	vboost_lvl	–	0 to 7	Defines the voltage boost level.
	iboost_en	–	0 to 1	Enables the output current boost mode.
	iboost_lvl	–	0 to 15	Defines the current boost level.
	tx_eq_pre	–	0 to 63	Defines the transmit pre-equalizer setting.
	tx_eq_main	–	0 to 63	Defines the transmit main equalizer setting.
	tx_eq_post	–	0 to 63	rx_los_thr
Receive Path				
	rx_slos_thr	–	0 to 7	Defines the SerDes threshold for detection of a loss of signal condition.
	rx_adapt_cont	–	0 to 1	Enable continuous adaptation
	rx_adapt_afe_en	–	0 to 1	Enable AFE processing
	rx_adapt_dfe_en	–	0 to 1	Enable DFE processing
	rx_eq_attn_lvl	–	0 to 7	Equalizer configuration
	rx_eq_ctle_boost	–	0 to 15	
	rx_eq_ctle_pole	–	0 to 7	
	rx_eq_vga1_gain	–	0 to 15	
	rx_eq_vga2_gain	–	0 to 15	
	rx_eq_dfe_tap1	–	0 to 256	

Table 12 PON SerDes Configuration Examples Values – ITU TDM PON Modes

Option	Application		
	ITU-T G.984 G-PON asymmetrical	ITU-T G.987 XG-PON asymmetrical	ITU-T G.9807 XGS-PON symmetrical
SFP – Transmit Path Parameters			
vboost_en	0	1	1
vboost_lvl	0	3	3
iboost_en	1	1	1
iboost_lvl	15	15	15
tx_eq_pre	0	0	0
tx_eq_main	2	2	0
tx_eq_post	10	10	29
SFP – Receive Path Parameters			
rx_slos_thr	7	7	7
rx_adapt_afe_en	0	0	0
rx_adapt_dfe_en	0	0	0
rx_adapt_cont	0	0	0
rx_eq_attn_lvl	7	7	7
rx_eq_ctle_boost	0	0	0
rx_eq_ctle_pole	6	6	6
rx_eq_vga1_gain	5	5	5
rx_eq_vga2_gain	5	5	5
rx_eq_dfe_tap1	0	0	0
SFU – Transmit Path Parameters			
vboost_en	0	1	1
vboost_lvl	0	8	8
iboost_en	1	1	1
iboost_lvl	15	15	15
tx_eq_pre	0	0	17
tx_eq_main	4	4	36
tx_eq_post	13	13	5
SFU – Receive Path Parameters			
rx_slos_thr	7	7	7
rx_slos_thr	7	7	7
rx_adapt_afe_en	0	0	0
rx_adapt_dfe_en	0	0	0
rx_adapt_cont	0	0	0
rx_eq_attn_lvl	7	7	7
rx_eq_ctle_boost	0	0	0

Table 12 PON SerDes Configuration Examples Values – ITU TDM PON Modes (cont'd)

Option	Application		
	ITU-T G.984 G-PON asymmetrical	**ITU-T G.987 XG-PON asymmetrical**	**ITU-T G.9807 XGS-PON symmetrical**
rx_eq_ctle_pole	6	6	6
rx_eq_vga1_gain	5	5	5
rx_eq_vga2_gain	5	5	5

Table 13 PON SerDes Configuration Examples Values – ITU TWDM PON Modes

Option	Application	
	ITU-T G.989 NG-PON2 symmetrical	**ITU-T G.989 NG-PON2 asymmetrical**
SFP – Transmit Path Parameters		
vboost_en	1	1
vboost_lvl	3	3
iboost_en	1	1
iboost_lvl	15	15
tx_eq_pre	0	0
tx_eq_main	0	2
tx_eq_post	29	10
SFP – Receive Path Parameters		
rx_slos_thr	7	7
SFU – Transmit Path Parameters		
vboost_en	1	1
vboost_lvl	8	8
iboost_en	1	1
iboost_lvl	15	15
tx_eq_pre	17	0
tx_eq_main	36	4
tx_eq_post	5	13
SFU – Receive Path Parameters		
rx_slos_thr	7	7

PON SerDes Configuration Database

For known optical interfaces, a PON SerDes-related database is provided with the software. Upon the first start of the software (first boot), the board hardware type is checked against the database for a matching entry.

- When an entry is found, the contents are copied into the `/etc/serdes/config` file.
- When no matching entry is found, the default values are used.

The database is a simple directory, holding a single file for each known optical interface type. The file names are constructed as: `<boardname>.conf`.

For example:

```
# ll /rom/etc/serdes-db
drwxr-xr-x    2 root     root           293 May 8 14:08 ./
drwxr-xr-x   21 root     root           956 May 8 14:16 ../
-rw-r--r--    1 root     root           219 May 8 11:47 default.conf
-rw-r--r--    1 root     root           220 May 8 11:47 prx120-sfu-qspi-pon.conf
-rw-r--r--    1 root     root           221 May 8 11:47 prx126-sfp-eva-pon.conf
-rw-r--r--    1 root     root           221 May 8 11:47 prx126-sfp-pon.conf
-rw-r--r--    1 root     root           221 May 8 11:47 prx321-eva-pon.conf
-rw-r--r--    1 root     root           221 May 8 11:47 prx321-eva-qspi-pon.conf
-rw-r--r--    1 root     root           220 May 8 11:47 prx321-sfu-qspi-pon.conf
-rw-r--r--    1 root     root           220 May 8 11:47 prx321-sfu-spi-pon.conf
-rw-r--r--    1 root     root           222 May 8 11:47 urx851-eva.conf
-rw-r--r--    1 root     root           222 May 8 11:47 urx851-ref.conf
```

This initialization is done only once upon first boot. After successful initialization, this script is removed to reduce the start-up time.

The recommended way to adopt other PCB designs is to add related configuration data files to the database folder.

2.4.5 PON Interface Initialization

The PON interface is composed of the internal, on-chip PON hardware (PON IP) and the external PMD and BOSA components.

2.4.5.1 PON IP Initialization

The PON IP hardware module is controlled by the PON firmware. This firmware is loaded by the OMCI daemon (omcid). The firmware binary files are located at `/lib/firmware`.

```
# ll /lib/firmware/
drwxr-xr-x  3 root root           209 Apr 2 13:27 ./
drwxr-xr-x 10 root root           467 Apr 2 13:27 ../
-rw-r--r--  1 root root        187384 Apr 2 13:27 intel_gpon_fw_b.bin
-rw-r--r--  1 root root        211948 Apr 2 13:27 intel_xpon_fw_b.bin
-rw-r--r--  1 root root        118980 Apr 2 13:27 ltq_fw_PHY31G_IP_prx3xx_Bxx.bin
-rw-r--r--  1 root root        211512 Apr 2 13:27 ppv4-qos-fw.elf
-rw-r--r--  1 root root          2436 Apr 2 13:27 prx_i2c_aca_fw.bin
drwxr-xr-x  2 root root           196 Apr 3 2020 voice/
```

The firmware files are related to this functionality.

- `intel_gpon_fw_b.bin`: PON interface in G-PON operation mode
- `intel_xpon_fw_b.bin`: PON interface in XG-PON or XGS-PON operation mode (or NG-PON2)
- `ppv4-qos-fw.elf`: Quality of Service (QoS) module
- `ltq_fw_PHY31G_IP_prx3xx_Bxx.bin`: On-chip 2.5G Ethernet PHY
- `prx_i2c_aca_fw.bin`: I²C slave interface (for SFP applications based on PRX126 only)
- `voice/*`: VoIP codec firmware

Note: The firmware update residing in external components (such as a third party external Ethernet PHY) is not covered by the software. This must be handled by the third party software delivered by the respective component vendor.

2.4.5.2 PMD Initialization

The external PMD device is accessed through an I²C serial interface. Depending on the device type, it may contain an integrated or external EEPROM device to store calibration and configuration data. If no EEPROM hardware is used with the PMD device, the calibration and configuration data must be stored into the PMD upon each power cycle through the software. This kind of initialization is not part of the 10G PON Chipset System Package and must be added by the ONU system vendor.

The access is handled by the PON mailbox driver (pon_mbox), providing virtual files. The path to these EEPROM devices is defined in the optical interface configuration file `/etc/config/optic`. See **Chapter 2.4.3.8**.

If there is no EEPROM hardware available to store non-volatile PMD data, a file must be created and the UCI configuration shall point to this file instead of a physical EEPROM device. This allows to apply the same software handling for PMD identification, even if a physical EEPROM device is not present.

2.4.6 GPIO Handling

The GPIO pins used by the application must be defined in the device tree according to the ONU hardware implementation. This section provides some examples of the GPIO handling in the device tree and driver code.

Note: The code snippets shown below are not related to each other, check the code base for details.

GPIO Example Code 1

Define a GPIO pin in the device tree to make it available to the driver. In this case, it is done in the file prx126-sfp.dtsi. In this example, the physical pin GPIO8 is defined to be used as MOD_DEF0.

```
/** I2C Slave emulation with EEPROM */
&sfp_i2c {
        status = "okay";
        gpio-moddef0 = <&gpio0 8 GPIO_ACTIVE_HIGH>;
};
```

GPIO Example Code 2

The GPIO functions are called by the kernel drivers. This example shows how the MOD_DEF0 output pin is handled by the `pon_sfp_i2c` driver. The code example checks whether the GPIO used as MOD_DEF0 is valid (`gpio_is_valid`). When valid, it checks whether the pin is requested (`devm_gpio_request`).

Note: This is done because the pin is defined as optional. When it is defined as required, the check is not required.

```
priv->gpio_moddef0 = of_get_named_gpio(dev->of_node, "gpio-moddef0", 0);
if (gpio_is_valid(priv->gpio_moddef0)) {
        ret = devm_gpio_request(dev, priv->gpio_moddef0, "moddef0");
        if (ret) {
                dev_err(dev, "failed to request gpio %d\n",
                        priv->gpio_moddef0);
                return ret;
        }
}
```


GPIO Code Example 3

When the pin function is available, it is possible to enable its output function as shown in the example. When the GPIO is requested, the pin is switched to input mode and the external pull-up device provides a high signal level.

```
drv_pon_i2c_sfp_processing_enable(struct pon_sfp_i2c_priv *priv, bool enable)
{
        pon_sfp_i2c_wr32(ACA_FW_IFC_OFFSET(enable), !!enable,
                         priv->base.genrisc_spram);

        /* Indicate availability of SFP on enable */
        if (gpio_is_valid(priv->gpio_moddef0)) {
                if (enable)
                        gpio_direction_output(priv->gpio_moddef0, 0);
                else
                        gpio_direction_input(priv->gpio_moddef0);
        }
        return 0;
}
```

When there is no external pull-up device and to enable the GPIO output driver, use the command: `gpio_direction_output(priv->gpio_moddef0, 1);`

2.4.6.1 GPIO Initialization for SFP

For SFP applications, the usage of GPIO signals towards the SFP connector is application-specific. Beside the default SFP interface functions, it is possible to enable additional functionality, for example to provide ToD information. Use the DTS file (`prx126-sfp.dtsi`) to select which GPIO pins to use for specific SFP functions.

For example:

```
&pinctrl {
        rx_los_disabled: rx-los-disabled {
                intel,groups = "rx_los_dis";
                intel,function = "gpio";
                intel,pins = <25>;
                intel,mux = <EQBR_MUX_0>;
        };

        rx_los_enabled: rx-los-enabled {
                intel,groups = "rx_los_en";
                intel,function = "pon_ip";
                intel,pins = <25>;
                intel,mux = <EQBR_MUX_2>;
        };

        pin_1pps_disabled: pin_1pps-disabled {
                intel,groups = "1pps_dis";
                intel,function = "gpio";
                intel,pins = <29>;
                intel,mux = <EQBR_MUX_0>;
        };

        ntr_disabled: ntr-disabled {
                intel,groups = "ntr_dis";
                intel,function = "gpio";
                intel,pins = <28>;
                intel,mux = <EQBR_MUX_0>;
        };

        ntr_enabled: ntr-enabled {
                intel,groups = "ntr_en";
                intel,function = "pon_ip";
                intel,pins = <28>;
                intel,mux = <EQBR_MUX_2>;
        };
};
```


With these definitions, use the PON library function `fapi_pon_pin_config_set` to configure the individual functions. The available selections in this function are defined by enumerations.

To select the target pin, use:

```
/** GPIO port ID */
enum pon_gpio_pin_id {
        /** RX_LOS pin */
        PON_GPIO_PIN_ID_RX_LOS = 0,
        /** 1PPS pin */
        PON_GPIO_PIN_ID_1PPS = 1,
        /** NTR pin */
        PON_GPIO_PIN_ID_NTR = 2
};
```

while for the individual setting, use:

```
/** GPIO port status */
enum pon_gpio_pin_status {
        /** Use default settings for pin */
        PON_GPIO_PIN_STATUS_DEFAULT = 0,
        /** Set pin to disable state */
        PON_GPIO_PIN_STATUS_DISABLE = 1,
        /** Set pin to enable state */
        PON_GPIO_PIN_STATUS_ENABLE = 2
};
```


2.4.6.1.1 I²C Slave Bus Multiplexer for SFP

This chapter applies only to ONU applications integrated within an SFP/SFP+ form factor package, based on PRX126 devices. See **Chapter 10** for details.

These applications support an I²C slave interface emulation to allow host access according to the SFP and SFP+ specifications **[67]**. This interface is available after the operating system boot and full initialization.

To allow host access to the I²C interface of the SFP immediately after inserting the SFP into the host system, it is recommended to include a hardware multiplexer (dual low-impedance analog switch). **Figure 3** shows this setup.

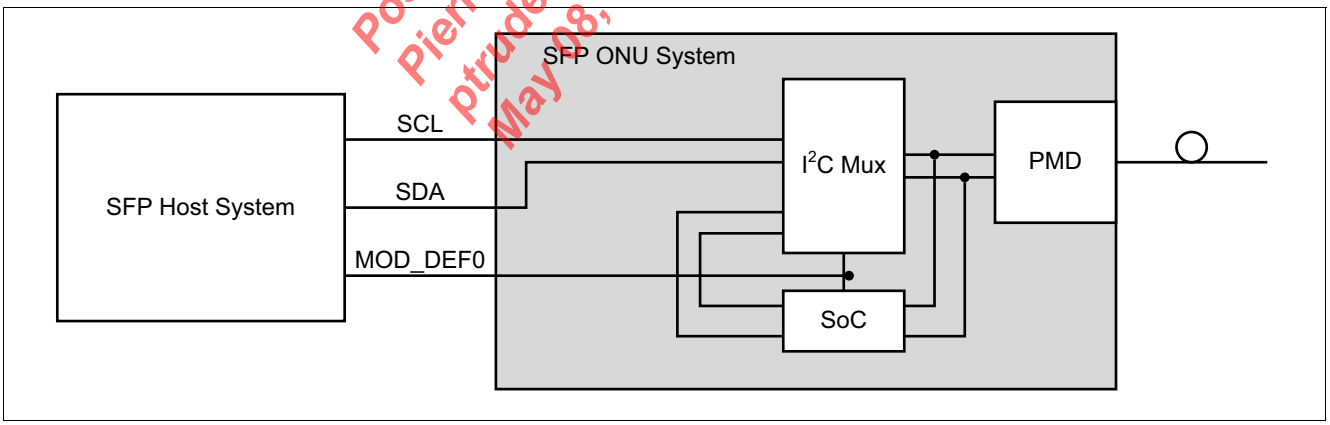


Figure 3 I²C Slave Bus Hardware Multiplexer

The multiplexer connects the I²C host interface to the PMD device until the system software is loaded and the I²C slave emulation is ready. This is signaled through the MOD_DEF0 signal to the host and this signal is also used to switch the I²C bus multiplexer. The reference design uses GPIO8 for this signal.

The related device tree setting in `prx126-sfp.dtsi` is:

```
/** I2C Slave emulation with EEPROM */
&sfp_i2c {
        status = "okay";
        gpio-moddef0 = <&gpio0 8 GPIO_ACTIVE_HIGH>;
};
```

2.4.6.1.2 I²C Master Bus Multiplexer

When more I²C devices with identical addresses are present in the application, an external I²C bus multiplexer may be required. The target bus leaf is selected through the I²C bus itself (see **Figure 4**). Such devices (PCA954x or similar) are supported and show up in the boot log, when present. For example:

```
[    9.480180] i2c-lantiq 16340000.i2c: version 1.01
[    9.485097] i2c i2c-0: Added multiplexed i2c bus 3
[    9.489109] i2c i2c-0: Added multiplexed i2c bus 4
[    9.493629] i2c i2c-0: Added multiplexed i2c bus 5
[    9.498377] i2c i2c-0: Added multiplexed i2c bus 6
[    9.502828] pca954x 0-0070: registered 4 multiplexed buses for I2C switch pca9545
...
[   10.052731] at24 3-0050: 256 byte 24c02 EEPROM, writable, 1 bytes/write
[   10.058829] at24 3-0051: 256 byte 24c02 EEPROM, writable, 1 bytes/write
[   10.065587] at24 4-0050: 256 byte 24c02 EEPROM, writable, 1 bytes/write
[   10.071994] at24 4-0051: 256 byte 24c02 EEPROM, writable, 1 bytes/write
```

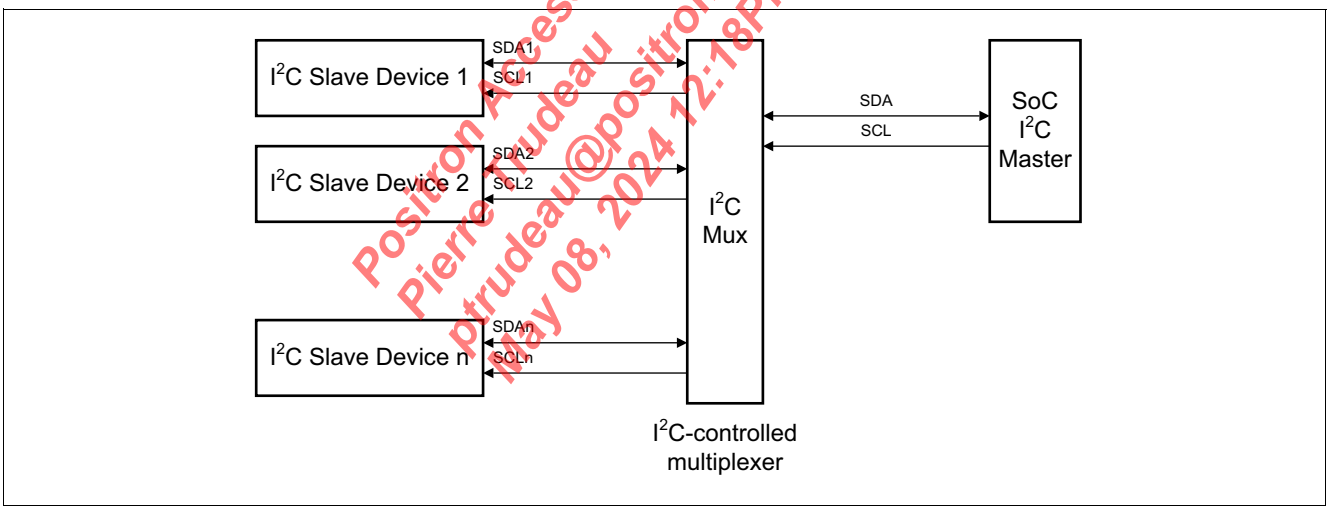



Figure 4 I²C Bus Multiplexer

2.4.7 Packet Buffer Configuration

The internal and external data buffer memory configuration influences the achievable data rate, latency, and power consumption. This must be adapted to the application targets.

The packet storage in the external DRAM memory uses four buffer pools to handle large and small packets separately and to isolate upstream from downstream traffic. The target is to provide an optimized DRAM buffer utilization while decoupling the upstream and downstream traffic flows.

The setting depends on the target application requirements and may involve vendor-specific adaptation or fine-tuning.

This architecture is recommended as a default configuration:

- Assign buffer pools 0 and 1 exclusively for downstream data traffic.
- Assign buffer pools 2 and 3 exclusively for upstream data traffic.
- Use buffer pools 0 and 2 for smaller data packets (below 704 bytes).
- Use buffer pools 1 and 3 for larger data packets (705 to 9920 bytes).

Table 14 shows the recommended values.

Table 14 Packet Buffer Pool Configuration Values

Buffer Pool	Direction	Packet Size Range	Buffer Size	Number of Buffers
0	Downstream	Up to 704 bytes	1024 bytes	17536
1	Downstream	705 to 9920 bytes	10240 bytes	1664
2	Upstream	Up to 704 bytes	1024 bytes	78080
3	Upstream	705 to 9920 bytes	10240 bytes	6784

The configuration is done in the device tree file `prx300.dtsi`, for example:

```
/* Total number of buffers in pools 0-3 */
intel,bm-buff-num = <0x4480 0x680 0x13100 0x1a80>;
/*                    17536   1664   78080   6784 */

/* Size of each buffer in pools 0-3*/
intel,bm-buff-size = <0x400 0x2800 0x400 0x2800>;
/*                     1024  10240  1024  10240 */

/* Split the buffer pools into 2 for DS and 2 for US */
intel,bm-buff-split = <0x2 0x2>;
```

According to the selected buffer configuration, the available queue sizes must be defined in the OMCI MIB initialization file.

When the whole buffer space must be used for upstream and downstream traffic without dedicated reservation, the setting must be:

```
/* Buffer pools are shared for downstream and upstream (not split). */
intel,bm-buff-split = <0x0 0x0>;
```

Settings other than `<0x0 0x0>` and `<0x2 0x2>` are not supported.

2.4.8 External Ethernet PHY Configuration

The handling of external Ethernet PHYs is not covered by the SDK, this must be checked with the PHY vendor.

2.5 Data Rate and Packet Rate Limitation

It is possible to limit the maximum amount of data forwarded within a given time interval. The limitation is achieved in two basic ways, by shaping and by policing (also called metering).

While shaping limits the output of a queue to a configured maximum bit rate, policing measures the packet rate at a given point in the data flow and drops all packets in excess of the configured data rate. For this reason, a policer always leads to packet loss while a shaper only loses data when the related queue runs full and rejects to accept more data packets. On the other hand, shaping always requires a queue and the according memory resources.

2.6 Performance Monitoring Counter Handling

The performance counters are provided by several parts of the hardware. The main blocks are the PON IP, the Ethernet MACs, and the Gigabit Ethernet switch IP (GSWIP). Whenever possible the ethtool functions are used to retrieve the counter values, otherwise dedicated driver and library functions are provided to access these counters. Other counters implemented by the Linux operating system are not covered here.

2.6.1 PON Counters

The PON counters are accessed through the PON library (pon_lib) functions.

- `fapi_pon_alloc_counters_get`
- `fapi_pon_alloc_discard_counters_get`
- `fapi_pon_eth_rx_counters_get`
- `fapi_pon_eth_tx_counters_get`
- `fapi_pon_fec_counters_get`
- `fapi_pon_gem_port_counters_get`
- `fapi_pon_gtc_counters_get`
- `fapi_pon_ploam_ds_counters_get`
- `fapi_pon_ploam_us_counters_get`
- `fapi_pon_psm_counters_get`
- `fapi_pon_xgtc_counters_get`

Each of these counters is defined as `uint64_t` and rolls over when the maximum counter value is reached. The higher layer software must handle such rare overflow events. There is no counter reset function provided to avoid memory overhead for counters which may not be used by the application.

For an implementation example, refer to the OMCI stack example code which is part of the 10G PON Chipset System Package delivery.

For counter details, refer to the Doxygen HTML documentation derived from the source code. The control file is located at `/ponlib/doc/doxyconfig`.

Note: Some counters are available in debug mode only, when the secure debug feature is enabled.

2.6.2 Ethernet Switch Counters

The switch API provides a set of RMON counters accessible through `GSW_RMON_PORT_GET`. The counter structure uses `u32` elements, except for the byte-based counters implemented as `u64` (`nRxGoodBytes`, `nTxGoodBytes`). Given this, counter overflow events are rare and handled by the higher layer software.

For details, refer to the Doxygen HTML documentation derived from the source code.

Several counters are also available through the ethtool function. For example:

```
# ethtool -S eth0_0
NIC statistics:
     RxExtendedVlanDiscardPkts: 0
     MtuExceedDiscardPkts: 0
     TxUnderSizeGoodPkts: 0
     TxOversizeGoodPkts: 0
     RxGoodPkts: 104317
     RxUnicastPkts: 14474
     RxBroadcastPkts: 44338
     RxMulticastPkts: 45505
     RxFCSErrorPkts: 0
     RxUnderSizeGoodPkts: 0
     RxOversizeGoodPkts: 0
     RxUnderSizeErrorPkts: 0
     RxGoodPausePkts: 0
     RxOversizeErrorPkts: 0
     RxAlignErrorPkts: 0
     RxFilteredPkts: 0
     Rx64BytePkts: 54378
     Rx127BytePkts: 38434
     Rx255BytePkts: 5330
     Rx511BytePkts: 5858
     Rx1023BytePkts: 301
     RxMaxBytePkts: 16
     TxGoodPkts: 0
     TxUnicastPkts: 0
     TxBroadcastPkts: 0
     TxMulticastPkts: 0
     TxSingleCollCount: 0
     TxMultCollCount: 0
     TxLateCollCount: 0
     TxExcessCollCount: 0
     TxCollCount: 0
     TxPauseCount: 0
     Tx64BytePkts: 0
     Tx127BytePkts: 0
     Tx255BytePkts: 0
     Tx511BytePkts: 0
     Tx1023BytePkts: 0
     TxMaxBytePkts: 0
     TxDroppedPkts: 0
     TxAcmDroppedPkts: 0
     RxDroppedPkts: 0
     RxGoodBytes: 10710747
```

```
RxBadBytes: 0
TxGoodBytes: 0
RxOverflowError: 0
```

2.6.3 Ethernet Port Counters – ethtool

Ethernet counters are provided by the Linux ethtool function. This is called for physical interfaces but also for all internally defined network devices, such as GEM ports. For example:

```
ethtool -S eth0_0
ethtool -S gem1234
```

Not all counters are supported at each of the interfaces and also not always in both directions (rx and tx). For example, FCS/CRC error counters are only available at physical interfaces in ingress direction.

2.6.4 Counter Overflow Handling

The worst case (minimum) time taken into account for counter overflow handling is calculated as follows.

2.6.4.1 64-bit Byte Counter Overflow

The number of bytes possibly handled on a 10G interface increments at a maximum rate of $r_{byte} = 1.25 * 10^9\ s^{-1}$.

A 64-bit counter rolls over at a count of $cnt_{64} = 2^{64} = 1.845 * 10^{19}$.

The shortest roll-over time is calculated as $t_{min64} = cnt_{64}/r_{byte} = 1.845 * 10^{19}/1.25 * 10^9\ s^{-1} = 1.476 * 10^{10}$ s = 467 a.

2.6.4.2 32-bit Packet Counter Overflow

The number of packets (shortest Ethernet frames) possibly handled on a 10G interface increments at a maximum rate of less than $r_{packet} = 1.25 * 10^9\ s^{-1}/64 = 1.95 * 10^7$ (not taking the IPG or GEM/XGEM headers into account, which relaxes this further).

A 32-bit counter rolls over at a count of $cnt_{32} = 2^{32} = 4.29 * 10^9$.

The shortest roll-over time is calculated as $t_{min32} = cnt_{32}/r_{packet} = 4.29 * 10^9/1.95 * 10^7\ s^{-1} = 220$ s = 3 min 40 s.

2.7 Inter-Process Communication

The data exchange between individual software processes uses the `ubus` method for Inter-Process Communication (IPC). See **[38]** for a general reference.

The `ubus` provides the communication channel between individual software components that run in different processes.

There are several cases implemented that leverage this interface:

- Software upgrade
- PON status information retrieval
- IP Host network interface configuration

2.7.1 IPC for Software Upgrade

The software upgrade daemon communicates with the OMCI daemon to handle OMCI-controlled software upgrade. Check the `pon_img_lib` (PON Image Library) sources for more detail.

The usage can be deactivated if the target system shall us a different IPC method. This selection is up to the system software implementer.

2.7.2 IPC for PON Status Information

On a prplOS-based system, the PON HAL uses the `ubus` to retrieve PON status information from the OMCI daemon. The supported data model is defined in **[33]**.

The usage of this is optional and deactivated by default for non-prplOS image types.

2.8 Under-voltage Detection

A hardware comparator is used to detect if the application's main supply voltage drops below a threshold and generates an interrupt. For example, a nominal supply voltage of 12 V is fed to the under-voltage detection input through a resistive voltage divider to trigger an interrupt if the voltage drops below about 10 V. This leaves some time to save important data into flash memory and to inform other parts of the network.

The time left after an under-voltage event, also known as "dying gasp" event, is determined by the application's hardware design (blocking capacitance versus current drawn by the application).

2.8.1 Under-voltage Detection Configuration

The interrupt is available as #46 in the MIPS interrupt controller.

To use it, it must be defined in the device tree as

```
interrupt-parent = <&gic>;
interrupts = <GIC_SHARED 46 IRQ_TYPE_LEVEL_HIGH>;
```

This can in turn be registered by a driver that handles the interrupt. The driver must make use of the device tree settings.

To check the behavior of the under-voltage detection, the raw interrupt status registers can be dumped, for example:

Under-voltage interrupt inactive:

```
root@prx321-sfu-qspi-pon:/# io -4 -l32 0x12320480
12320480:  00000000 00000000 00000000 00000000
12320490:  00000000 00000000 00000000 00000000
```

Under-voltage interrupt active:

```
root@prx321-sfu-qspi-pon:/# io -4 -l32 0x12320480
12320480:  00000000 00004000 00000000 00000000
12320490:  00000000 00000000 00000000 00000000
```


2.8.2 Under-voltage Detection Handling

While an under-voltage state is detected, the PON module starts sending the "dying gasp" information to the OLT.

If the power rises above the detection threshold instead of going further down, the PON module stops sending the "dying gasp" information and resumes normal operation.

Due to the fact that the duration of such an under-voltage condition and the lowest power level reached are unknown, the best practice is to reset the system by the software and ensure that the power drop does not lead to an instable or unpredictable system state.

2.9 Interrupt Handling

Interrupts are provided by the MIPS interrupt controller. Table 15 provides a list of the available interrupt sources and the numbering that must be used to register a specific interrupt source.

Table 15 Interrupt Numbering

Interrupt Number	Interrupt Short Name	Interrupt Source
0	MPS_IR9	MPS Status Interrupt #0 (Linux VPE to Voice VPE)
1	MPS_IR10	MPS Status Interrupt #1 (Linux VPE to Voice VPE)
2	MPS_IR11	MPS Status Interrupt #2 (Linux VPE to Voice VPE)
3	MPS_IR12	MPS Status Interrupt #3 (Linux VPE to Voice VPE)
4	MPS_IR13	MPS Status Interrupt #4 (Linux VPE to Voice VPE)
5	MPS_IR14	MPS Status Interrupt #5 (Linux VPE to Voice VPE)
6	MPS_IR15	MPS Status Interrupt #6 (Linux VPE to Voice VPE)
7	MPS_IR16	MPS Semaphore Interrupt to Voice VPE
8	MPS_IR17	MPS Global Interrupt to Voice VPE
9	PCIe_1_INTA	PCIe Core 1 Interrupt A
10	PCIe_1_INTB	PCIe Core 1 Interrupt B
11	PCIe_1_INTC	PCIe Core 1 Interrupt C
12	PCIe_1_INTD	PCIe Core 1 Interrupt D
13	DMA_FCC_INT	DMA FCC Interrupt
14	SPI_RIR	SPI 0 Receive Interrupt
15	SPI_TIR	SPI 0 Transmit Interrupt
16	SPI_EIR	SPI 0 Error Interrupt
17	SPI_FIR	SPI 0 Frame Interrupt
18	–	Reserved, not used.
19	TOH_ERR	Temperature Overheating Interrupt
20	I2C2_FIFO_IR	I2C2 FIFO Error Interrupt
21	I2C1_FIFO_IR	I2C1 FIFO Error Interrupt
22	DDR_DFI_ALERT	DDR DFI Alert Interrupt
23	I2C_FIFO_IR	I2C FIFO Error Interrupt
24	LAN_XPCS0_IR	LAN XPCS 0 Interrupt
25	LAN_XPCS1_IR	LAN XPCS 1 Interrupt
26	WAN_XPCS_IR	WAN XPCS Interrupt
27	–	Reserved, not used.
28	–	Reserved, not used.
29	ACA_HOST_IR	ACA GENRISC Host Upper Interrupt
30	MACsec_IG_IR	MACsec Ingress Interrupt
31	MACsec_EG_IR	MACsec Egress Interrupt
32	I2C_BREQ_IR	I2C Burst Request Interrupt
33	I2C_LBRE_IR	I2C Last Burst Request Interrupt
34	I2C_SREQ_IR	I2C Single Request Interrupt

Table 15 Interrupt Numbering (cont'd)

Interrupt Number	Interrupt Short Name	Interrupt Source
35	I2C_LSRE_IR	I2C Last Single Request Interrupt
36	I2C_P_IR	I2C Protocol Interrupt
37	MSI_1_IR0	MSI Core 1 Interrupt 0
38	MSI_1_IR1	MSI Core 1 Interrupt 1
39	MSI_1_IR2	MSI Core 1 Interrupt 2
40	MSI_1_IR3	MSI Core 1 Interrupt 3
41	MSI_1_IR4	MSI Core 1 Interrupt 4
42	MSI_1_IR5	MSI Core 1 Interrupt 5
43	MSI_1_IR6	MSI Core 1 Interrupt 6
44	MSI_1_IR7	MSI Core 1 Interrupt 7
45	GSWIPSS_IR	GSWIP Subsystem Interrupt
46	DGASP_IR	Under-voltage (Dying Gasp) Interrupt
47	–	Reserved, not used.
48	LAN_SW_INT	Gigabit LAN Switch (Core) Interrupt
49	PCIe_1_IR	PCIe Core 1 Legacy Interrupt
50	–	Reserved, not used.
51	PCM_TXIR	PCM Transmit Interrupt
52	PCM_RXIR	PCM Receive Interrupt
53	–	Reserved, not used.
54	–	Reserved, not used.
55	–	Reserved, not used.
56	IPI0	IPI Software Interrupts
57	IPI1	
58	IPI2	
59	IPI3	
60	IPI4	
61	IPI5	
62	IPI6	
63	IPI7	
64	DMA0_IR	DMA0 All Channel Interrupt
65	DMA1T_IR	DMA1 TX All Channel Interrupt
66	DMA1R_IR	DMA1 RX All Channel Interrupt
67	DMA2T_IR	DMA2 TX All Channel Interrupt
68	DMA2R_IR	DMA2 RX All Channel Interrupt
69	–	Reserved, not used.
70	DMA4_IR	DMA4 All Channel Interrupt
71	HOLD_IR	DPLL Hold Signal Interrupt
72	FLOCK_IR	DPLL FLOCK Signal Interrupt

Table 15 Interrupt Numbering (cont'd)

Interrupt Number	Interrupt Short Name	Interrupt Source
73	PHASE_LOCK_IR	DPLL Phase Lock Signal Interrupt
74	PLL3_LOCK_IR	PLL3 Lock Signal Interrupt
75	PLL5_LOCK_IR	PLL5 Lock Signal Interrupt
76	ASC1_TIR	ASC 1 Transmit Interrupt
77	ASC1_TBIR	ASC 1 Transmit Buffer Interrupt
78	ASC1_RIR	ASC 1 Receive Interrupt
79	ASC1_EIR	ASC 1 Error Interrupt
80	ASC1_ABSTIR	ASC 1 Autobaud Start Interrupt
81	ASC1_SFCIR	ASC1 Software Flow Control Interrupt
82	ASC1_ABDETIR	ASC 1 Autobaud Detection Interrupt
83	P2_ERR	P2 Clock Supervision Interrupt
84	PCM_TX_EOP	DMA FCC PCM TX End of Packet Interrupt
85	–	Reserved, not used.
86	PCM_RX_EOP	DMA FCC PCM RX End of Packet Interrupt
87	PON_HOST_IBIR	PON Host Mailbox Input Interrupt
88	PON_HOST_OBIR	PON Host Mailbox Output Interrupt
89	–	Reserved, not used.
90	PON_HOST_IR	PON Host Interrupt
91	–	Reserved, not used.
92	REFCLK_REQ_IR	WAN PHY Reference Clock Request Interrupt
93	–	Reserved, not used.
94	PMI_IR	Interconnect Performance Monitor Interrupt
95	8_KHZ	8-kHz Interrupt sourced from PLL1
96	–	Reserved, not used.
97	SPI0_Compl	SPI 0 True Complete Interrupt
98	SPI1_Compl	SPI 1 True Complete Interrupt
99	SPI1_RIR	SPI 1 Receive Interrupt
100	SPI1_TIR	SPI 1 Transmit Interrupt
101	SPI1_EIR	SPI 1 Error Interrupt
102	SPI1_FIR	SPI 1 Frame Interrupt
103	ASC_TIR	ASC 0 Transmit Interrupt
104	ASC_TBIR	ASC 0 Transmit Buffer Interrupt
105	ASC_RIR	ASC 0 Receive Interrupt
106	ASC_EIR	ASC 0 Error Interrupt
107	ASC_ABSTIR	ASC 0 Autobaud Start Interrupt
108	ASC_ABDETIR	ASC 0 Autobaud Detection Interrupt
109	ASC_SFCIR	ASC 0 Software Flow Control Interrupt
110	QSPI_IR	QSPI Slave Response Interrupt

Table 15 Interrupt Numbering (cont'd)

Interrupt Number	Interrupt Short Name	Interrupt Source
111	NGNA_BM	NGNA Buffer Manager Interrupts (all)
112	NGNA_QOS	NGNA QoS Engine Interrupts (all)
113	–	Reserved, not used.
114	EIP_123_VPE0	EIP123 Crypto Engine Interrupts
115	EIP_123_VPE1	
116	EIP_123_VPE2	
117	EIP_123_VPE3	
118	TC1A	General Purpose Timer/Counter Interrupts
119	TC1B	
120	TC2A	
121	TC2B	
122	TC3A	
123	TC3B	
124	–	Reserved, not used.
125	–	Reserved, not used.
126	GPHY_INT	GPHY Interrupt
127	GPHYCDB_INT	GPHY CDB Interrupt
128	MSI_0_IR0	MSI Core 0 Interrupts
129	MSI_0_IR1	
130	MSI_0_IR2	
131	MSI_0_IR3	
132	MSI_0_IR4	
133	MSI_0_IR5	
134	MSI_0_IR6	
135	MSI_0_IR7	
136	PCIe_0_INTA	PCIe Core 0 Interrupts
137	PCIe_0_INTB	
138	PCIe_0_INTC	
139	PCIe_0_INTD	
140	CM2_ERR_INT	CM2 Error Interrupt
141	CM2_PM_INT	CM2 Performance Counter Interrupt

Table 15 Interrupt Numbering (cont'd)

Interrupt Number	Interrupt Short Name	Interrupt Source
142	MPS_IR0	MPS Status Interrupts (Voice VPE to Linux VPE)
143	MPS_IR1	
144	MPS_IR2	
145	MPS_IR3	
146	MPS_IR4	
147	MPS_IR5	
148	MPS_IR6	
149	MPS_IR7	
150	MPS_IR8	
151	SSX1_ERR_IR0	SSX1 Error Interrupt
152	PCIe_0_IR	PCIe Core 0 Legacy Interrupt
153	SSX1_ERR_IR1	SSX1 PM Error Interrupt
154	–	Reserved, not used.
155	–	Reserved, not used.
156	–	Reserved, not used.
157	–	Reserved, not used.
158	–	Reserved, not used.
159	SSX7_Error	Cross Bar SSX7 Error Interrupt
160	TC1_1A	Timer/Counter Interrupts
161	TC1_1B	
162	TC1_2A	
163	TC1_2B	
164	TC1_3A	
165	TC1_3B	
166	TC2_1A	
167	TC2_1B	
168	TC2_2A	
169	TC2_2B	
170	TC2_3A	
171	TC2_3B	
172	GPIO_INT0IR	GPIO 0-31 Interrupt
173	GPIO_INT1IR	GPIO 32-63 Interrupt
174	–	Reserved, not used.
175	–	Reserved, not used.
176	–	Reserved, not used.
177	–	Reserved, not used.
178	–	Reserved, not used.
179	–	Reserved, not used.

Table 15 Interrupt Numbering (cont'd)

Interrupt Number	Interrupt Short Name	Interrupt Source
180	–	Reserved, not used.
181	–	Reserved, not used.
182	I2C1_BREQ_IR	I2C1 Interrupts
183	I2C1_LBRE_IR	
184	I2C1_SREQ_IR	
185	I2C1_LSRE_IR	
186	I2C1_P_IR	
187	I2C2_BREQ_IR	I2C2 Interrupts
188	I2C2_LBRE_IR	
189	I2C2_SREQ_IR	
190	I2C2_LSRE_IR	
191	I2C2_P_IR	
192	CBM_IR_0	CBM (Buffer Manager) Interrupts
193	CBM_IR_1	
194	CBM_IR_2	
195	CBM_IR_3	
196	CBM_IR_4	
197	CBM_IR_5	
198	CBM_IR_6	
199	CBM_IR_7	
200	IPI8	IPI Software Interrupts
201	IPI9	
202	IPI10	
203	IPI11	
204	IPI12	
205	IPI13	
206	–	Reserved, not used.
207	–	Reserved, not used.
208	–	Reserved, not used.
209	MB_INT	V-CODEC Event and Error Interrupt
210	MB_INT_IB_0	V-CODEC Command Inbox Interrupt
211	MB_INT_IB_1	V-CODEC Data Inbox Interrupt
212	MB_INT_OB_0	V-CODEC Command Outbox Interrupt
213	MB_INT_OB_1	V-CODEC Data Outbox Interrupt
214	MPE_MCPY_ERR	MCPY Module inside MPE Interrupt

Table 15 Interrupt Numbering (cont'd)

Interrupt Number	Interrupt Short Name	Interrupt Source
215	MPS2_IR0	MPS Status Interrupts (M4Kec to InterAptiv cluster)
216	MPS2_IR1	
217	MPS2_IR2	
218	MPS2_IR3	
219	MPS2_IR4	
220	MPS2_IR5	
221	MPS2_IR6	
222	MPS2_IR7	
223	MPS2_IR8	
224	MPE_REQ_0_IR	MPE Request Interrupts
225	MPE_REQ_1_IR	
226	MPE_REQ_2_IR	
227	MPE_REQ_3_IR	
228	MPE_REQ_4_IR	
229	MPE_REQ_5_IR	
230	MPE_REQ_6_IR	
231	MPE_REQ_7_IR	
232	MPE_REQ_8_IR	
233	MPE_REQ_9_IR	
234	MPE_REQ_10_IR	
235	MPE_REQ_11_IR	
236	MPE_RES_0_IR	MPE Response Interrupts
237	MPE_RES_1_IR	
238	MPE_RES_2_IR	
239	MPE_RES_3_IR	
240	MPE_RES_4_IR	
241	MPE_RES_5_IR	
242	MPE_RES_6_IR	
243	MPE_RES_7_IR	
244	MPE_RES_8_IR	
245	MPE_RES_9_IR	
246	MPE_RES_10_IR	
247	MPE_RES_11_IR	

Table 15 Interrupt Numbering (cont'd)

Interrupt Number	Interrupt Short Name	Interrupt Source
248	MPE_MCPY_P0_IR	MPE Mcpy Interrupts
249	MPE_MCPY_P1_IR	
250	MPE_MCPY_P2_IR	
251	MPE_MCPY_P3_IR	
252	MPE_MCPY_P4_IR	
253	MPE_MCPY_P5_IR	
254	MPE_MCPY_P6_IR	
255	MPE_MCPY_P7_IR	

3 Networking Configuration

This chapter describes the networking configuration aspects of

- Interface network devices
- Bridges
- Extended VLAN handling
- Quality of Service (QoS)
- Multicast packet forwarding

3.1 Interface Network Devices

This is an example of network devices in the ONU which are visible to the Linux stack. Table 16 describes these network devices in more detail.

```
# ip -br l
lo               UNKNOWN  00:00:00:00:00:00 <LOOPBACK,UP,LOWER_UP>
ins0             UNKNOWN  00:00:00:00:00:00 <BROADCAST,MULTICAST,NOARP,UP,LOWER_UP>
dummy0           UNKNOWN  02:89:d1:15:57:30 <BROADCAST,NOARP,UP,LOWER_UP>
teql0            DOWN     <NOARP>
eth0_0           DOWN     00:e0:92:00:01:44 <BROADCAST,MULTICAST>
eth0_0_1_lct     UNKNOWN  00:e0:92:00:01:40 <BROADCAST,MULTICAST,UP,LOWER_UP>
eth0_0_2         DOWN     00:e0:92:00:01:44 <BROADCAST,MULTICAST>
eth0_0_3         DOWN     00:e0:92:00:01:44 <BROADCAST,MULTICAST>
eth0_0_us        DOWN     b6:62:be:c5:1a:64 <BROADCAST,MULTICAST>
eth0_1           DOWN     00:e0:92:00:01:45 <BROADCAST,MULTICAST>
eth0_1_1_lct     DOWN     ee:a6:ac:cc:cb:7c <BROADCAST,MULTICAST>
eth0_1_2         DOWN     00:e0:92:00:01:45 <BROADCAST,MULTICAST>
eth0_1_3         DOWN     00:e0:92:00:01:45 <BROADCAST,MULTICAST>
eth0_1_us        DOWN     3a:6d:ab:d2:d9:1c <BROADCAST,MULTICAST>
pon0             DOWN     00:00:00:00:00:00 <BROADCAST,MULTICAST>
tcont-omci@pon0  DOWN     32:2c:da:53:39:c4 <BROADCAST,MULTICAST,M-DOWN>
gem-omci@pon0    UNKNOWN  00:e0:92:00:01:42 <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN>
iphost1@pon0     DOWN     4e:ee:ba:c8:a9:eb <BROADCAST,MULTICAST,M-DOWN>
```


Table 16 Network Devices in ONU

Interface Name	Interface Type	Description
eth0_0	LAN port 1	LAN/UNI port network device created by the Ethernet driver
eth0_0_1_lct		Local debug access
eth0_0_2		Additional hardware bridge ports for multicast handling
eth0_0_3		Additional hardware bridge ports for broadcast handling
eth0_1	LAN port 2	LAN/UNI port network device created by the Ethernet driver. Applicable only when the second interface is active.
eth0_1_1_lct		Local debug access
eth0_1_2		Additional hardware bridge ports for multicast handling
eth0_1_3		Additional hardware bridge ports for broadcast handling
pon0	PON port	PON port network device created upon insertion of the G-PON data path driver
gem-omci	OMCC GEM for management	ONU management interface where the OMCI packets are exchanged. This corresponds to CTP index 0.
tcont-omci		

Table 16 Network Devices in ONU (cont'd)

Interface Name	Interface Type	Description
pmapper<X>	Multi-GEM IEEE 802.1p mapper	When a group of GEM, such as multi-GEM, is used, this group is called pmapper.
gem<X>	PON channel (GEM)	Logical PON channel, such as a GEM port
tcont<X>	T-CONT upstream network device	Upstream T-CONT network device
iphost1	IP Host or IPv6 Host	Network interface for VoIP data transfer

Various configuration using standard Linux utilities, extended utilities or proprietary commands are issued on these set of network devices.

See **Chapter 11.4** for differences in an HGU application.

3.1.1 Interface MAC Addresses

MAC addresses are assigned to interfaces during the system start-up procedure. The default assignment is defined in the script `pon.sh` (see **Chapter 4.2.3.2**).

Up to six MAC addresses are assigned incrementally starting from the base MAC address that is defined by the U-Boot environment variable `ethaddr`.

Enhance or modify the script according to the application requirements.

```
pon_mac_get() {
        local mac_offset
        local mac_limit=6

        case "$1" in
        eth0_0_1_lct | eth0_1_1_lct | lct)
                mac_offset=0
                ;;
        host | wan | eth1)
                mac_offset=1
                ;;
        eth0_0 | eth0_0_[23])
                # subifs 2/3 are for MC/BC and can use same mac as main ifc
                mac_offset=2
                ;;
        eth0_1 | eth0_1_[23])
                # subifs 2/3 are for MC/BC and can use same mac as main ifc
                mac_offset=3
                ;;
        iphost)
                mac_offset=$((4+$2))
                ;;
        *)
                mac_offset=-1
                ;;
        esac

        if [ $mac_offset -ge 0 ]; then
                echo $(_mac_add_offset $(pon_base_mac_get) $mac_offset)
        fi
```

```
        if [ "$mac_offset" -ge "$mac_limit" ]; then
                echo "pon.sh[warning]: used mac_offset($mac_offset) over the
limit($mac_limit)" >&2
        fi
}
```

3.1.2 UNI Interface Naming

On PRX120 based systems there is only a single UNI port available, using the integrated Ethernet PHY. This is named `eth0_0` in the default images.

The default images created for PRX321 offer two UNI ports. The internal PHY interface is named `eth0_1` while the external PHY interface is `eth0_0`.

3.2 Bridges

The transparent bridges based upon destination MAC address based forwarding of packets are the basic building block of any Layer 2 modem. The system software uses standard Linux bridges, wherever it is deemed appropriate. **Figure 5** and **Figure 6** show the typical bridges.

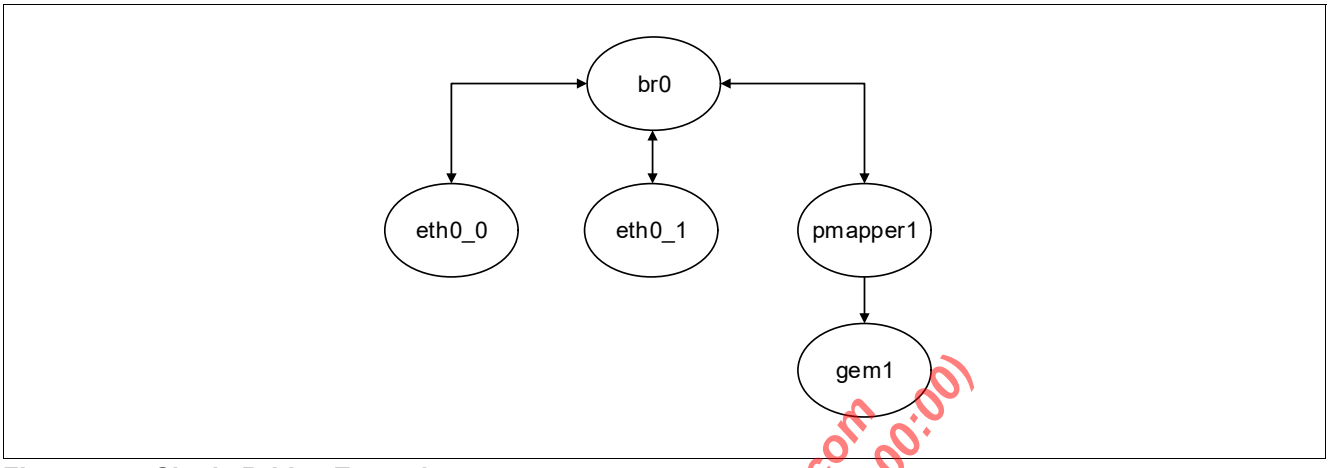


Figure 5 Single Bridge Example

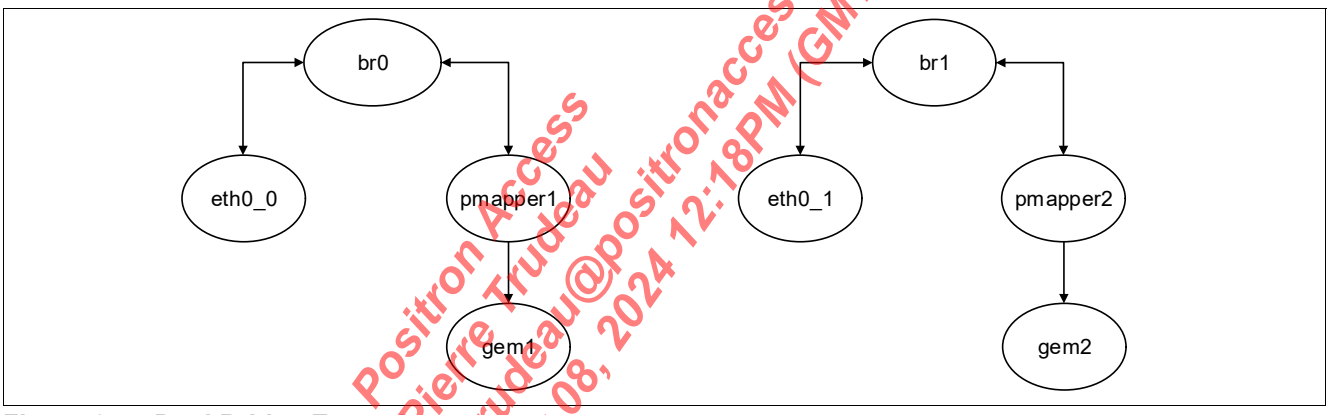


Figure 6 Dual Bridge Example

The bridges are created using Linux commands, such as `brctl`, `ip` or `bridge` command line utilities. The Linux bridges are offloaded to the hardware switch by the Data Path Manager (DP Mgr) driver. The driver implements the necessary callback operations provided by the Linux `switchdev` infrastructure to enable this functionality.

Attention: Bridges created outside the OMCI must use different names to prevent them being removed from the OMCI stack.

Bridge Aging Time

Although it is possible to instantiate multiple bridges for hardware acceleration, the aging time is a common configuration value. Different aging time values cannot be configured to individual hardware accelerated bridge instances.

3.3 Bridging Domains

The data path manager driver supports bridging domains. These domains allow to configure the set of bridge ports to which each bridge port is able to forward traffic to. The hardware bridge ports have a bridge port map that defines the possible destination bridge port set.

The bridging domains are used in the PON, LAN and VUNI Ethernet Drivers. The domains numbers are defined in the Linux kernel sources and located at `include/net/datapath_br_domain.h`.

When the bridging domains are not used, the bridge port map of each bridge port contains the bridge ports of all the network devices added to the software bridge. All network devices have a mapping to a bridge port in hardware. For example, using a bridge domain for the multicast GEM port allows the multicast data to be forwarded only to the UNI ports.

Table 17 Bridging Domain Mappings

Network Device	Domain	Port Map Forwarding Domain	Destination Network Devices
pmapper<n>	UCA(1)	UC(0) IPH(6) UCA(1)	eth0_0 eth0_1 iphost_bpX VUNI0_0
gem<n>	UCA(1)	UC(0) IPH(6) UCA(1)	eth0_0 eth0_1 iphost_bpX VUNI0_0
multicast gem	MC(2)	MC(2)	eth0_0_2 eth0_1_2 VUNI0_1
broadcast gem	BC1(3)	UC(0) BC2(4)	eth0_0_3 eth0_1_3 VUNI0_2
iphost_bp<n>	IPH(6)	IPH(6) UCA(1)	pmapper<n> gem<n>
eth0_0	UC(0)	UC(0) UCA(1)	pmapper<n> gem<n>
eth0_1	UC(0)	UC(0) UCA(1)	pmapper<n> gem<n>
eth0_0_1_lct	N/A	N/A	N/A
eth0_0_2	N/A	N/A	N/A
eth0_0_3	N/A	N/A	N/A
eth0_1_2	N/A	N/A	N/A
eth0_1_3	N/A	N/A	N/A
VUNI0_0	UC(0)	UC(0) UCA(1)	pmapper<n> gem<n>
VUNI0_1	N/A	N/A	N/A
VUNI0_2	N/A	N/A	N/A

Note: UCA(1) means unicast A domain and has numeric value of 1. Each bridge port map domain settings are shown using DPM driver debugfs. The Datapath driver uses a special flag for some ports “loop_dis”. This

flags disable loops between pon device bridge ports, for example pmappers cannot forward traffic to other pmappers. The settings can be checked in the datapath driver debugfs for each hardware logical port.

3.4 QoS Driver Architecture

The QoS configuration from the higher application layers is done using the standard Linux `tc qdiscs` and filters interface based on the kernel `rtnetlink` sockets. The QoS driver designed to cover the PON specific use cases offloads to the hardware. The driver is located in the Linux source directory at `drivers/net/datapath/pon_qos`. The driver translates the `tc` commands to the corresponding lower drivers APIs.

Figure 7 shows the main modules in the driver and the kernel functions being called when a `tc` configuration is done.

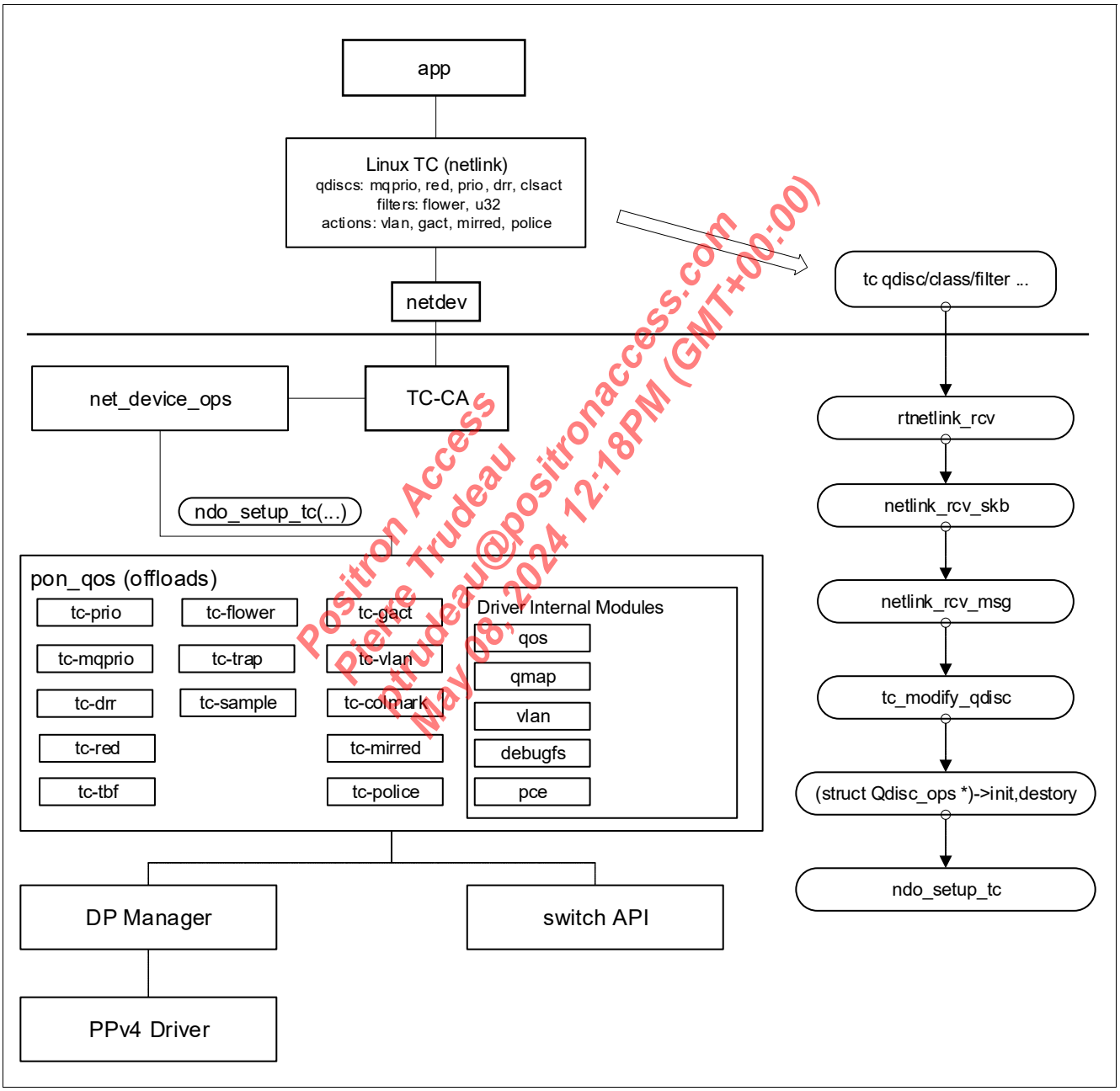


Figure 7 QoS Driver Architecture

The general driver structure is divided into two module categories, “General” and “Offload”.

General Modules

- `qos` - port, scheduler and queue configuration
- `qmap` - queue mapping and traffic class assignment
- `vlan` - VLAN translation, filtering and extended VLAN operations (split over several files)
- pce - handling of the switch packet classification engine rules
- `debugfs` - debug information

Note: The debug file system (debugfs) must only be used in special debug-enabled software images. It is not intended and tested for productive use.

Offload Modules

The offload modules translate the `qdisc` or filter attribute to the lower driver APIs using functions from the general modules.

- `tc-prio` - SP scheduling configuration
- `tc-mqprio` - SP scheduling configuration
- `tc-drr` - WRR scheduling configuration
- `tc-red` - WRED queue configuration
- `tc-tbf` - shaper configuration
- `tc-flower` - traffic classification
- `tc-gact` - drop/forwarding/filtering of packets
- `tc-vlan` - VLAN operations
- `tc-colmark` - color marking configuration
- `tc-mirred` - flowed based forwarding configuration
- `tc-police` - meter/policer configuration
- `tc-trap` - trapping packets to the CPU
- `tc-sample` - PCE counters configuration

See **Section 3.5** for details regarding these APIs, together with some `tc` command examples.

3.5 VLAN Modification

The OMCI standard specifies VLAN filtering as well as extended VLAN treatment (rule-based at ingress or egress of the bridge).

VLAN filtering and extended treatment are configured through the `tc` utility. The `ndo_setup_tc()` function must be written to control the desired behavior of the switch configuration.

Figure 8 shows all the modules necessary for the configuration of extended VLAN rules in hardware. The QoS driver is responsible for parsing the netlink attributes received from user space over the *tc-netlink* interface. It receives all the information in the call to the `ndo_setup_tc` triggers by the `tc-flower` classifier. The `clsact` qdisc must be configured on all devices where extended VLAN handling is required. When `tc-flower` classifier with `tc-vlan` action is created, the traffic control (TC) - classifier action (CA) subsystem is informed and in turn the `net_device_ops` hook `ndo_setup_tc` is called. These are examples.

Example to drop all untagged ingress packets from LAN port 1:

```
tc filter add dev eth0_0 ingress protocol all flower skip_sw action drop
```

Example to drop all single tagged ingress packets from LAN port 1:

```
tc filter add dev eth0_0 ingress protocol 802.1q flower skip_sw action drop
```

Example to drop all double tagged ingress packets from LAN port 1:

```
tc filter add dev eth0_0 ingress protocol 802.1ad flower skip_sw action drop
```

Example to add a VLAN tag with VID 888 and TPID 0x88a8 to all untagged ingress packets from LAN port 1:

```
tc filter add dev eth0_0 ingress prio 1 protocol all flower skip_sw vlan push id 888
protocol 0x88a8
```

Example to remove the VLAN tag when VID = 888 on egress of single tagged packets from LAN port 1:

```
tc filter add dev eth0_0 egress prio 2 protocol 802.1q flower skip_sw vlan_id 888
action vlan pop
```

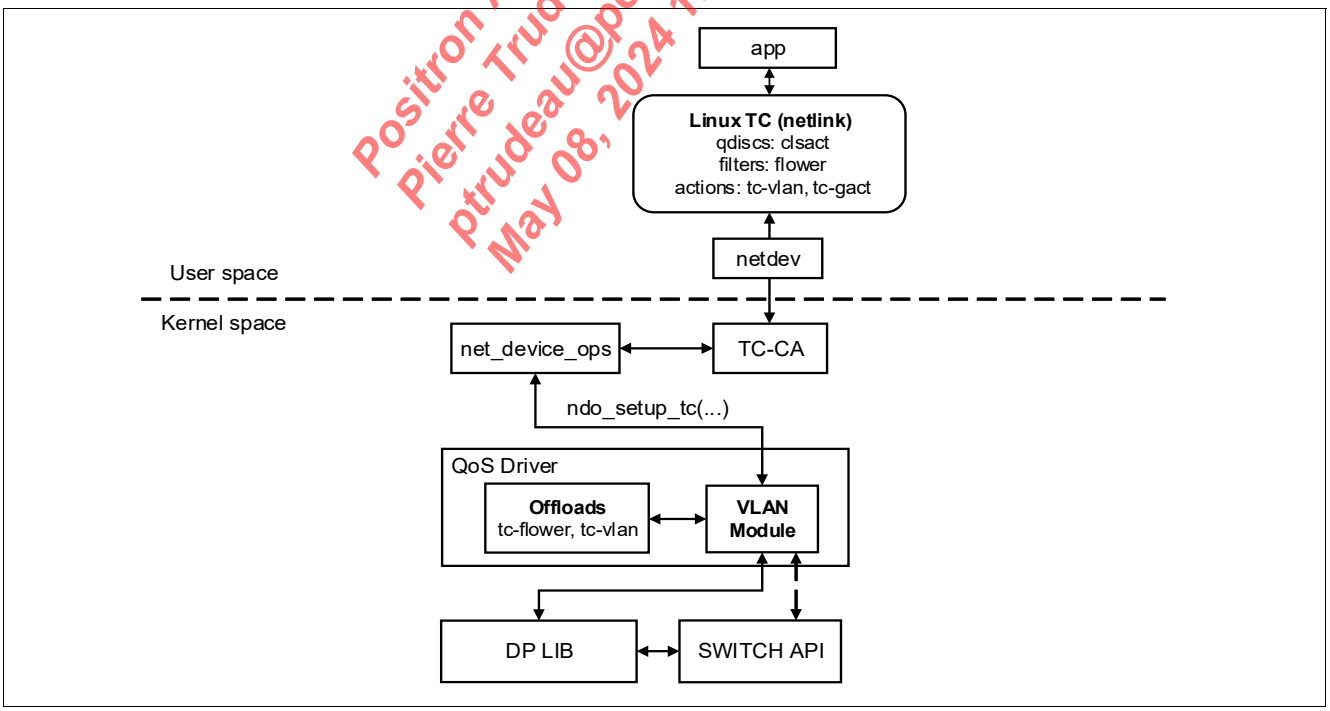



Figure 8 Software Modules Involved in Extended VLAN Configurations

OMCI VLAN Flow Handling

The OMCI VLAN flows require special handling in some cases. For example, in case of VLAN filtering based on the VLAN ID value, two `tc-flower` rules are required for protocol TPIDs `802.1q` and for `802.1ad`. These filter rules are configured on the `pmapper` network devices. The same is valid for the extended VLAN rules. When no filtering on inner or outer TPID is required by OMCI, several `tc-flower` rules are necessary. For single-tagged flows, two rules are required and for double-tagged flows, up to four rules.

It is possible to further increase the number of rules for cases when copying of the IEEE 802.1p priority bit or deriving the priority from DSCP is required.

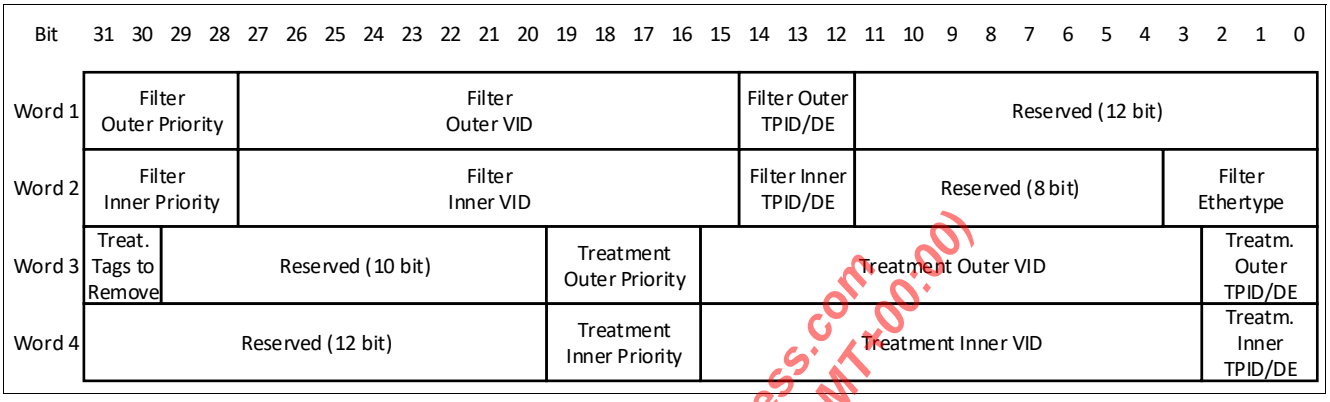


Figure 9 Received Frame VLAN Tagging Operation Table (Cookie)

VLAN Rule Sorting

The VLAN sorting is done in the QoS driver. The driver supports two sorting algorithms chosen during compile time. The OMCI standard sorting is enabled by default. The sorting functions are implemented in the `pon_qos_tc_vlan_storage.c` source file. Check for `pon_qos_tc_cookie_cmp<n>` with n = 0, 1, 2 for sorting untagged, single-tagged, and double-tagged rules respectively.

The sorting is based on the data passed in the cookie *tc-action* option. It contains all the information from the OMCI tagging operation table. Upstream as well as the resulting (inverse) downstream rule entries are sorted in the same manner.

The sorting algorithm is defined by the macro `cookie_cmp`. Any OLT-specific modifications to the sorting method must be applied here.

The OLT is identified by checking the **OLT Vendor ID** of the **OLT-G** OMCI managed entity (see **Section 4.20.6.2**). An example of such a check is found in the OMCI stack implementation (see `omci_olt_vendor_id_match` in `omci_extended_vlan_config_data.c`).

Usage of tc Action Cookie Options

The `tc-action` has a cookie option allowing to pass a 128-bit value not interpreted by the kernel. The OMCI stack uses the cookie to pass the **Received Frame VLAN Tagging Operation Table (Cookie)** information to the QoS driver to provide additional information about DEI bit handling and where it is possible to merge multiple `tc-flower` rules into a single rule in hardware. This is done when the VLAN P-bit is derived from DSCP, the P-bit or TPID is copied.

The QoS driver checks the FilterOuter TPID/DIE and FilterInner TPID/DEI options provided by the cookie and whether these options require filtering based on DEI, it applies the necessary configuration to hardware. The same applies for the TPID/DEI treatment.

The cookie also provides a hint that it is possible to merge multiple rules into one table rule in the switch hardware. For example, when we inserted a tag to a single tagged packet without filtering on the P-bit (`Filter Inner Priority = 8`) these `tc-flower` rules must cover this traffic flow:

```
tc filter add dev eth0_0 ingress pref 1 protocol 802.1q flower vlan_prio 0
vlan_ethtype arp action vlan push id 1000 priority 0 cookie $COOKIE
tc filter add dev eth0_0 ingress pref 2 protocol 802.1q flower vlan_prio 1
vlan_ethtype arp action vlan push id 1000 priority 1 cookie $COOKIE
tc filter add dev eth0_0 ingress pref 3 protocol 802.1q flower vlan_prio 2
vlan_ethtype arp action vlan push id 1000 priority 2 cookie $COOKIE
tc filter add dev eth0_0 ingress pref 4 protocol 802.1q flower vlan_prio 3
vlan_ethtype arp action vlan push id 1000 priority 3 cookie $COOKIE
tc filter add dev eth0_0 ingress pref 5 protocol 802.1q flower vlan_prio 4
vlan_ethtype arp action vlan push id 1000 priority 4 cookie $COOKIE
tc filter add dev eth0_0 ingress pref 6 protocol 802.1q flower vlan_prio 5
vlan_ethtype arp action vlan push id 1000 priority 5 cookie $COOKIE
tc filter add dev eth0_0 ingress pref 7 protocol 802.1q flower vlan_prio 6
vlan_ethtype arp action vlan push id 1000 priority 6 cookie $COOKIE
tc filter add dev eth0_0 ingress pref 8 protocol 802.1q flower vlan_prio 7
vlan_ethtype arp action vlan push id 1000 priority 7 cookie $COOKIE
```

The QoS driver sees the cookie has `Treatment Inner Priority equals 8` which means a VLAN tag must be added and its priority must be copied from the inner priority of the received frame. The driver evaluates this value and configures this operation to the switch hardware when the first rule is called by the user. After the rest of the rules are called, it checks whether their cookie matches a cookie in its internal list and does not configure any additional rule.

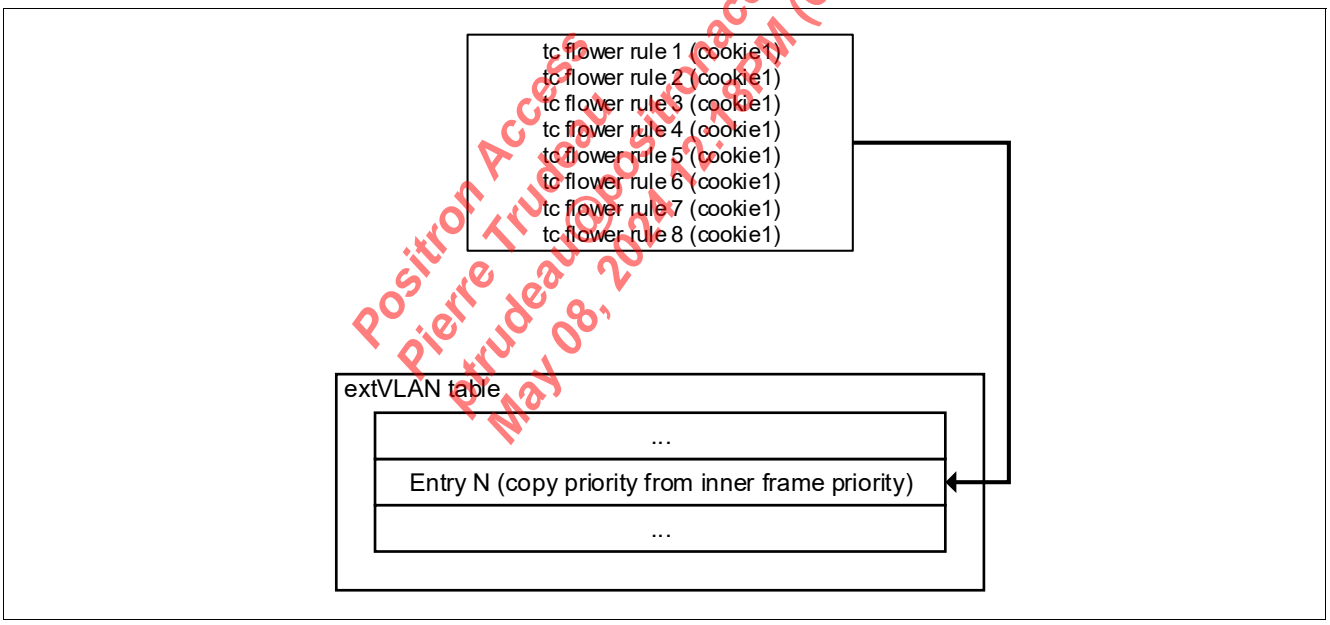


Figure 10 Mapping of Multiple tc flower Rules to a Single Rule in the Switch Table

In some cases, the drivers do the configuration after receiving these rules, when the DSCP to P-bit mapping rule is configured. Each rule must trigger a DSCP to P-bit mapping table update in the switch hardware.

1. `tc filter add dev $eth ingress pref 1 protocol ip flower ip_tos 0xe0/0xe0 action vlan push id 333 priority 7 protocol 802.1q cookie $COOKIE`
2. `tc filter add dev $eth ingress pref 2 protocol ip flower ip_tos 0xc0/0xc0 action vlan push id 333 priority 6 protocol 802.1q cookie $COOKIE`
3. `tc filter add dev $eth ingress pref 3 protocol ip flower ip_tos 0xa0/0xa0 action vlan push id 333 priority 5 protocol 802.1q cookie $COOKIE`
4. `tc filter add dev $eth ingress pref 4 protocol ip flower ip_tos 0x80/0x80 action vlan push id 333 priority 4 protocol 802.1q cookie $COOKIE`

5. `tc filter add dev $eth ingress pref 5 protocol ip flower ip_tos 0x60/0x60 action vlan push id 333 priority 3 protocol 802.1q cookie $COOKIE`
6. `tc filter add dev $eth ingress pref 6 protocol ip flower ip_tos 0x40/0x40 action vlan push id 333 priority 2 protocol 0x8100 cookie $COOKIE`
7. `tc filter add dev $eth ingress pref 6 protocol ip flower ip_tos 0x20/0x20 action vlan push id 333 priority 2 protocol 802.1q cookie $COOKIE`
8. `tc filter add dev $eth ingress pref 7 protocol ip flower ip_tos 0x0/0x0 action vlan push id 333 priority 1 protocol 0x8100 cookie $COOKIE`

All these rules trigger an update of the switch DSCP to P-bit mapping table. **Figure 11** depicts this process.

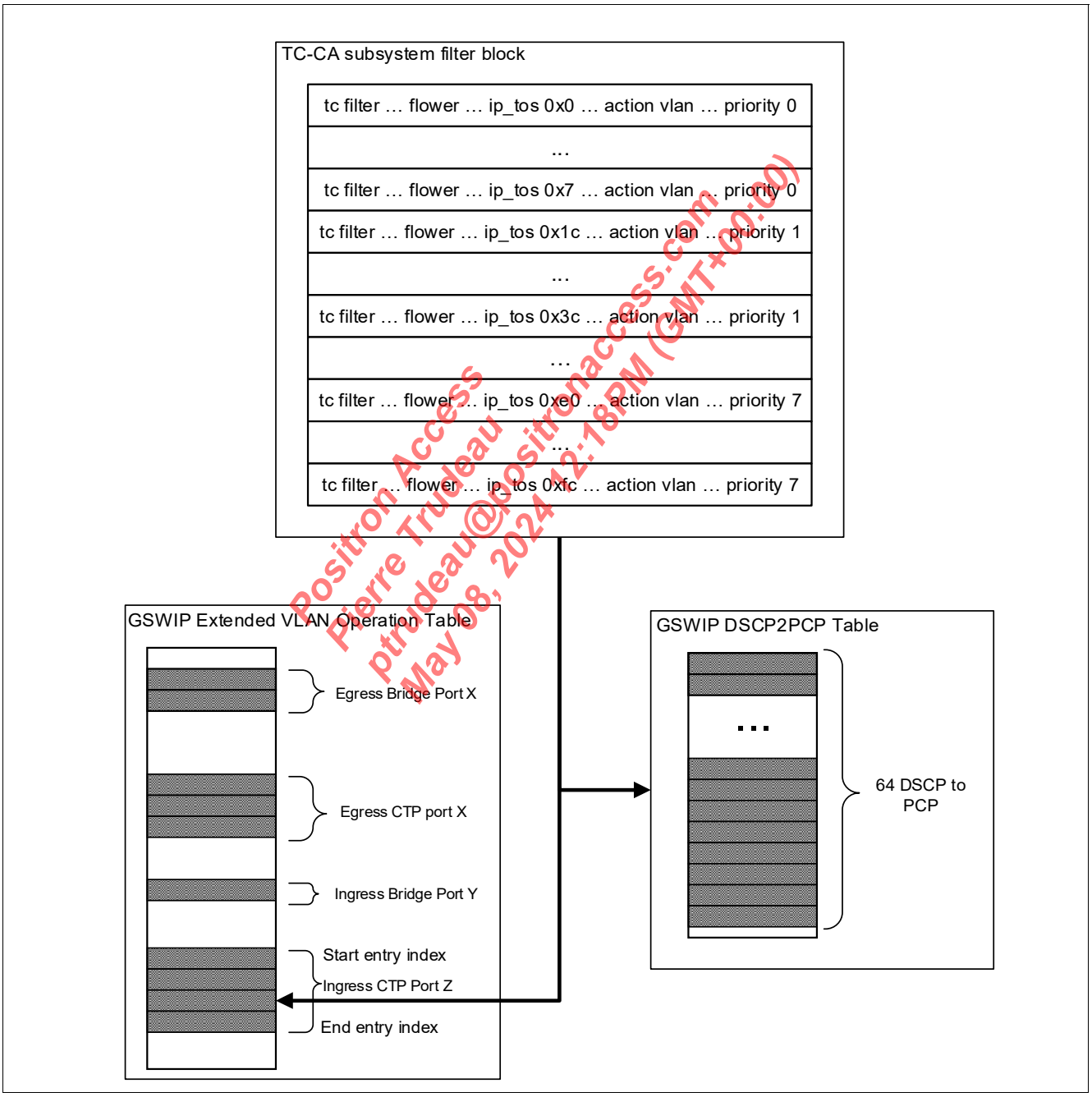


Figure 11 Offloading of tc flower ip_tos Rules to Switch Hardware

3.6 VLAN Filtering

The VLAN filtering allows to filter packets based on their VLAN. The packets are attached to bridge ports in both ingress and egress directions. Below there is an example of a simple VLAN filter on `802.1p pmapper ingress`:

```
tc filter add dev gem65534 ingress pref 65001 protocol all flower skip_sw indev
gem65534 action drop
tc filter add dev gem65534 ingress pref 1 protocol 802.1q flower skip_sw indev
gem65534  vlan_id 100 action pass
```

The first command adds a default rule that drops all packets and the second command adds a filter allowing only VLAN with `id 100` to pass through the bridge port of the `gem65534`. It is important to have both commands otherwise the hardware offload does not happen. The hardware table requires a default action for each filter. In this case, the `tc` rules result in a single hardware filter that pass VLAN with `id 100` and drops all the other packets. Using this set of default `tc` VLAN filtering rules achieves the same result:

```
tc filter add dev gem65534 ingress pref 3 protocol 802.1q flower skip_sw indev
gem65534 vlan_id 100 action ok
tc filter add dev gem65534 ingress pref 65001 protocol 802.1q flower skip_sw indev
gem65534 action drop
```

The default VLAN filtering rules start from `pref 65000`. The first rule uses a different protocol compared to the previous set of rules. Nevertheless, this results in the same hardware rule. The OMCI daemon configures several such default rules to match the Linux networking stack behaviour as close as possible. The `tc` filter commands must use the `flower indev` attribute specifying the same network device as the one used in the filter command. In the previous examples, this is `gem65534`. This instructs the driver to configure a VLAN filter in the hardware on the bridge port corresponding to the network device.

This is an example:

```
# tc filter show dev pmapper4354 ingress
filter protocol 802.1Q pref 1 flower
filter protocol 802.1Q pref 1 flower handle 0x74
  indev pmapper4354
  vlan_id 475
  skip_sw
        action order 1: gact action pass
         random type none pass val 0
         index 225 ref 1 bind 1

filter protocol 802.1ad pref 2 flower
filter protocol 802.1ad pref 2 flower handle 0x73
  indev pmapper4354
  vlan_id 475
  skip_sw
        action order 1: gact action pass
         random type none pass val 0
         index 227 ref 1 bind 1

filter protocol 802.1Q pref 3 flower
filter protocol 802.1Q pref 3 flower handle 0x72
  indev pmapper4354
  vlan_id 0
  skip_sw
        action order 1: gact action pass
         random type none pass val 0
```

```
         index 229 ref 1 bind 1

filter protocol 802.1ad pref 4 flower
filter protocol 802.1ad pref 4 flower handle 0x71
  indev pmapper4354
  vlan_id 0
  skip_sw
        action order 1: gact action pass
         random type none pass val 0
         index 231 ref 1 bind 1

filter protocol 802.1Q pref 65001 flower
filter protocol 802.1Q pref 65001 flower handle 0x70
  indev pmapper4354
  skip_sw
        action order 1: gact action drop
         random type none pass val 0
         index 233 ref 1 bind 1

filter protocol 802.1ad pref 65002 flower
filter protocol 802.1ad pref 65002 flower handle 0x6f
  indev pmapper4354
  skip_sw
        action order 1: gact action drop
         random type none pass val 0
         index 235 ref 1 bind 1

filter protocol all pref 65003 flower
filter protocol all pref 65003 flower handle 0x75
  indev pmapper4354
  skip_sw
        action order 1: gact action drop
         random type none pass val 0
         index 237 ref 1 bind 1
```


The VLAN filters action may have an action cookie value in which the complete VLAN `tci` and `tci_mask` are passed. It allows setting up the correct filter in the hardware using the correct DEI value as the `tc flower` has no option to specify classification based on the VLAN DEI bit. The cookie values match the filter values coming from the OMCI ME.

```
# DEI=0
filter protocol 802.1Q pref 9 flower
filter protocol 802.1Q pref 9 flower handle 0xa
  indev gem2201
  vlan_id 0
  vlan_prio 4
  skip_sw
       action order 1: gact action pass
        random type none pass val 0
        index 448 ref 1 bind 1
       cookie 8000ffff00000000000000000000000

# DEI=1
```

```
filter protocol 802.1Q pref 11 flower
filter protocol 802.1Q pref 11 flower handle 0xc
  indev gem2201
  vlan_id 0
  vlan_prio 4
  skip_sw
       action order 1: gact action pass
        random type none pass val 0
        index 452 ref 1 bind 1
       cookie 9000ffff0000000000000000000000000
```

3.7 Quality of Service

The implementation of the Quality of Service (QoS) feature in an ONU must take into consideration the UGW framework and leverage upon Linux offering.

The data path manager offers all QoS-related functions neutral to the underlying SoC hardware.

3.7.1 Upstream QoS

The upstream QoS is configured on the PON upstream port. The queues are added for each GEM on the attached T-CONT. Each of the T-CONT is visualized as a scheduler and up to 8 queues are attached to it with assigned priorities or weights.

- It is implemented by using the Linux `tc mqprio` for strict priority scheduling functions.
- The Linux `tc` function does not offer WRR or WFQ scheduling. The `tc` requires an extension to perform the WRR or WFQ configuration or the existing DRR scheduler is also used for this purpose.
- QoS-HAL offers a `ndo_tc_setup()` callback function such as `pp_qos_setup()` initialized to the PON T-CONT network device corresponding `ndo_tc_setup()` function.
- The TBF of `tc` is used to create traffic shapers.

Using mqprio to Configure Upstream Single Stage Strict Priority Scheduling

Figure 12 shows the qdisc hierarchy created by `mqprio`. The *mqprio* qdisc creates up to 16 queues depending on the user configuration. The default child qdisc type is set using `/proc/sys/net/core/default_qdisc`.

This command creates the setup on figure:

```
tc qdisc replace dev tcont1 root handle 8000:  \
        mqprio num_tc 8 map 0 1 2 3 4 5 6 7 \
        queues 1@0 1@1 1@2 1@3 1@4 1@5 1@6 1@7 hw 2

tc qdisc show dev tcont1
qdisc mqprio 8000: dev tcont1 root tc 8 map 0 1 2 3 4 5 6 7 0 0 0 0 0 0 0 0
             queues:(0:0) (1:1) (2:2) (3:3) (4:4) (5:5) (6:6) (7:7)
qdisc fq_codel 0: dev tcont1 parent 8000:8 limit 10240p flows 1024 quantum
1514 target 5.0ms interval 100.0ms memory_limit 32Mb ecn
qdisc fq_codel 0: dev tcont1 parent 8000:7 limit 10240p flows 1024 quantum 1514
target 5.0ms interval 100.0ms memory_limit 32Mb ecn
qdisc fq_codel 0: dev tcont1 parent 8000:6 limit 10240p flows 1024 quantum 1514
target 5.0ms interval 100.0ms memory_limit 32Mb ecn
qdisc fq_codel 0: dev tcont1 parent 8000:5 limit 10240p flows 1024 quantum 1514
target 5.0ms interval 100.0ms memory_limit 32Mb ecn
qdisc fq_codel 0: dev tcont1 parent 8000:4 limit 10240p flows 1024 quantum 1514
target 5.0ms interval 100.0ms memory_limit 32Mb ecn
qdisc fq_codel 0: dev tcont1 parent 8000:3 limit 10240p flows 1024 quantum 1514
target 5.0ms interval 100.0ms memory_limit 32Mb ecn
qdisc fq_codel 0: dev tcont1 parent 8000:2 limit 10240p flows 1024 quantum 1514
target 5.0ms interval 100.0ms memory_limit 32Mb ecn
qdisc fq_codel 0: dev tcont1 parent 8000:1 limit 10240p flows 1024 quantum 1514
target 5.0ms interval 100.0ms memory_limit 32Mb ecn
```

The command creates 8 queues with IDs ranging from 0x8001 to 0x8008 corresponding to priorities from 0 to 7. The value `hw = 2` specifies strict priority scheduling. The mqprio qdisc must be configured on the T-CONT network device which has multiple queues capable to allow the offload configuration.

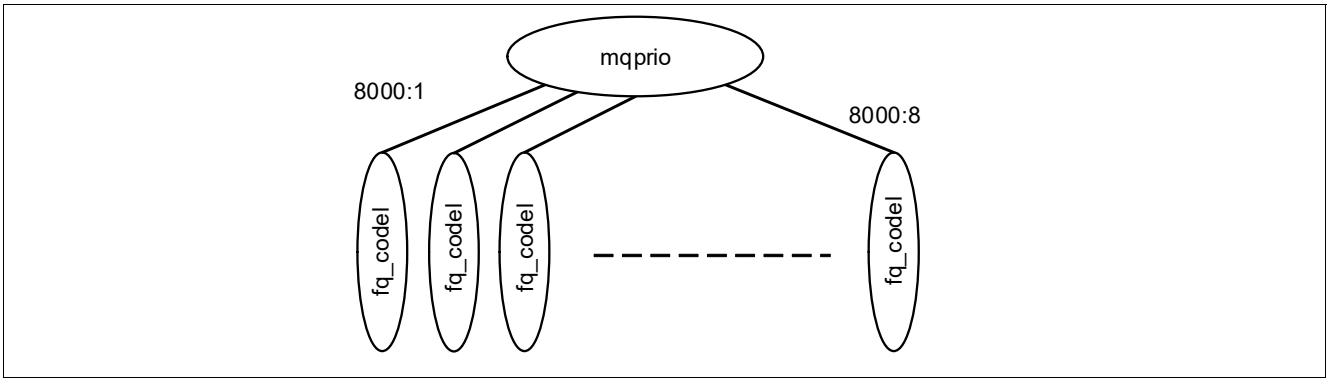


Figure 12 Linux mqprio qdisc Tree

Each of the mqprio qdiscs is replaceable by another one. For example, for the RED or WRED configuration, the qdisc is replaced by one or two `red` qdiscs respectively.

Use this command:

```
tc qdisc replace dev tcont1 parent 8000:1 handle 1001 \
         red limit 100k min 80k max 80k avpkt 1k burst 55 probability 1

tc qdisc show dev tcont1
qdisc mqprio 8000: root tc 8 map 0 1 2 3 4 5 6 7 0 0 0 0 0 0 0 0
             queues:(0:0) (1:1) (2:2) (3:3) (4:4) (5:5) (6:6) (7:7)
qdisc fq_codel 0: parent 8000:8 limit 10240p flows 1024 quantum 1514 target 5.0ms
interval 100.0ms memory_limit 32Mb ecn
qdisc fq_codel 0: parent 8000:7 limit 10240p flows 1024 quantum 1514 target 5.0ms
interval 100.0ms memory_limit 32Mb ecn
qdisc fq_codel 0: parent 8000:6 limit 10240p flows 1024 quantum 1514 target 5.0ms
interval 100.0ms memory_limit 32Mb ecn
qdisc fq_codel 0: parent 8000:5 limit 10240p flows 1024 quantum 1514 target 5.0ms
interval 100.0ms memory_limit 32Mb ecn
qdisc fq_codel 0: parent 8000:4 limit 10240p flows 1024 quantum 1514 target 5.0ms
interval 100.0ms memory_limit 32Mb ecn
qdisc fq_codel 0: parent 8000:3 limit 10240p flows 1024 quantum 1514 target 5.0ms
interval 100.0ms memory_limit 32Mb ecn
qdisc fq_codel 0: parent 8000:2 limit 10240p flows 1024 quantum 1514 target 5.0ms
interval 100.0ms memory_limit 32Mb ecn
qdisc red 1001: parent 8000:1 limit 100Kb min 30Kb max 80Kb
```


This replaces the child qdisc with ID 0x8000:1 with a red qdisc. **Figure 13** shows the configuration.

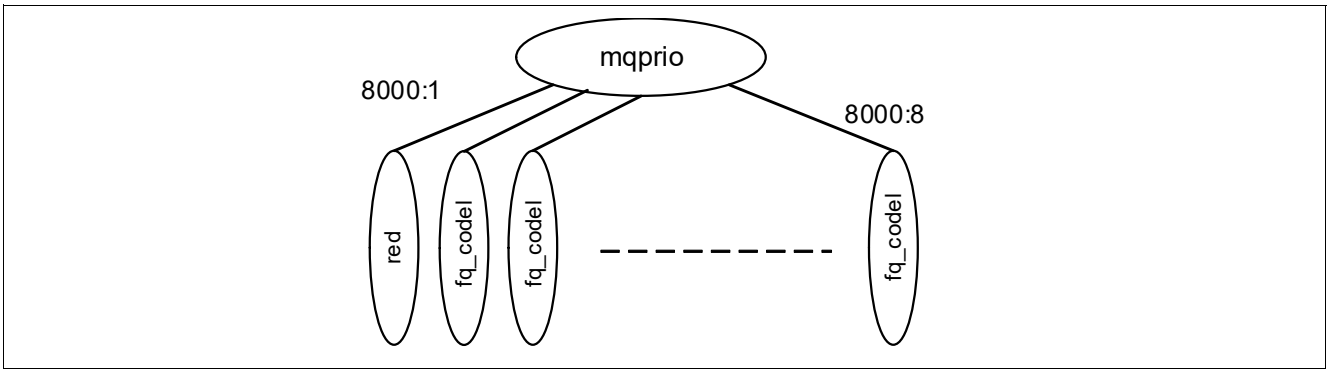


Figure 13 Linux mqprio child qdisc Replacement

The next step is to assign traffic class or GEM to queue and configure this mapping to hardware.

Use this command:

```
tc filter add dev t-cont3 ingress flower skip_sw indev gem7 classid 8000:5 action ok
```

This command assigns the GEM index 7 to mqprio child queue (qdisc) with ID 8000:5. In case of OMCI, these IDs must match the OMCI ME IDs. The hardware IDs are handled in the lower layers.

Figure 14 shows the complete mapping of the Linux qdiscs and network devices. The network devices must be provided by the corresponding data path driver.

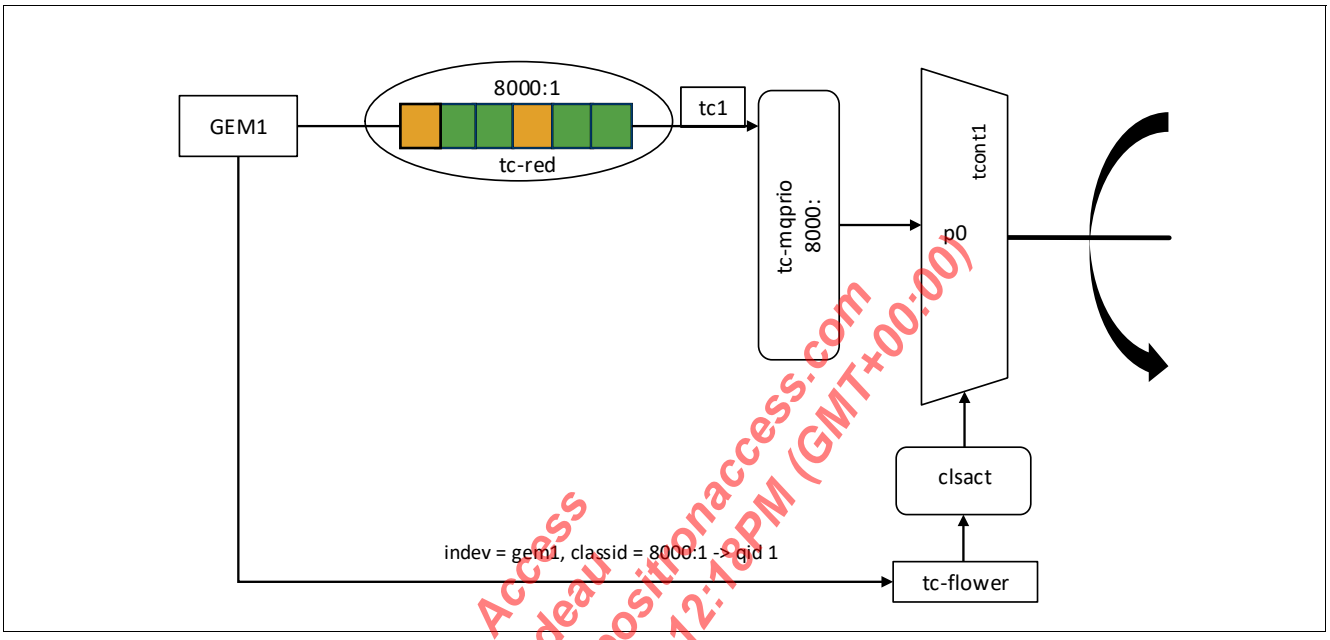


Figure 14 OMCI Single Stage Strict Priority Mapping to Linux qdiscs

Figure 15 shows the OMCI qdisc mapping with the qdisc and classifier relationships matching the OMCI traffic flow.

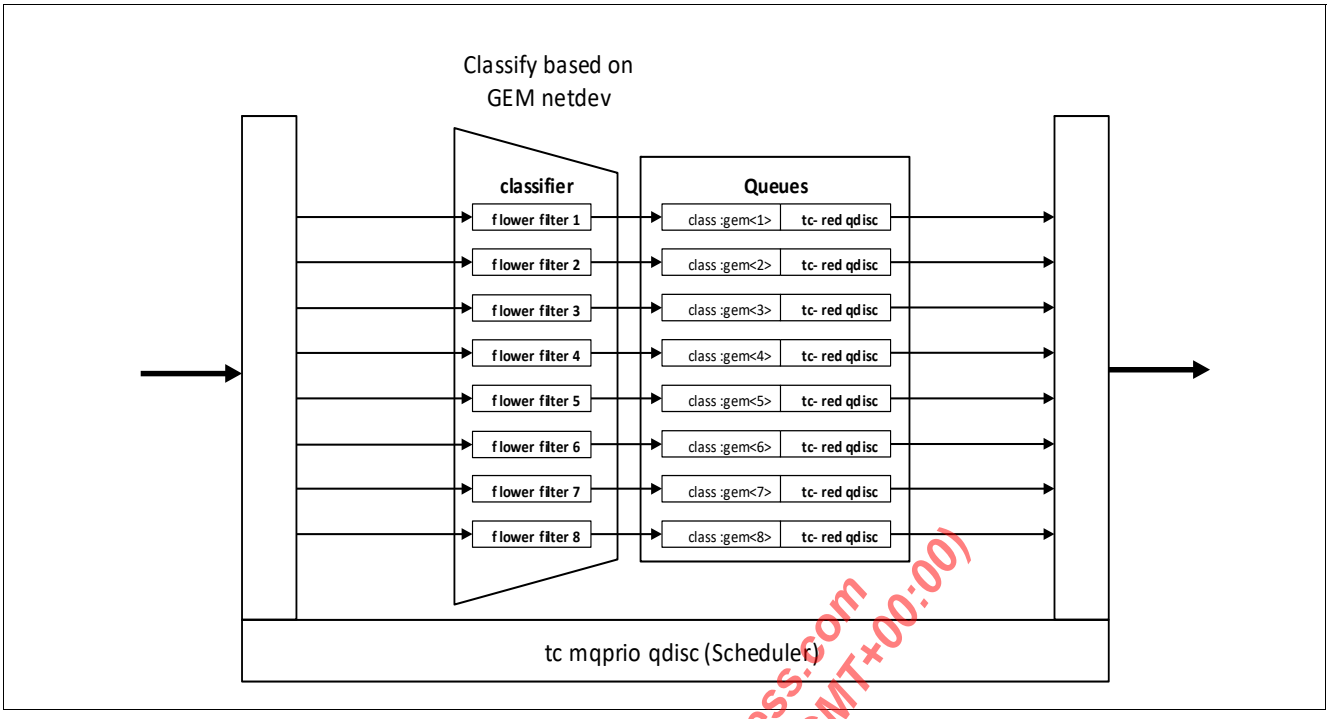


Figure 15 Linux mqprio qdisc Flow for OMCI

Figure 16 shows a complete configuration flow for assigning GEM to queue and the strict priority scheduler setup.

1. `tc qdisc replace dev tcont1 root handle 8000: mqprio num_tc 8 map 0 1 2 3 4 5 6 7 queues 1@0 1@1 1@2 1@3 1@4 1@5 1@6 1@7 hw 2`
2. GEM1:
   1. `tc qdisc replace dev tcont1 parent 8000:1 handle 1001 red limit 100k min 30k max 80k avpkt 1k burst 55 probability 1`
   2. `tc filter add dev tcont1 ingress flower skip_sw indev gem1 classid 8000:1 action ok`

   GEM2:
   1. `tc qdisc replace dev tcont1 parent 8000:2 handle 1002 red limit 100k min 30k max 80k avpkt 1k burst 55 probability 1`
   2. `tc filter add dev tcont1 ingress flower skip_sw indev gem2 classid 8000:2 action ok`

   GEM3:
   1. `tc qdisc replace dev tcont1 parent 8000:3 handle 1003 red limit 100k min 30k max 80k avpkt 1k burst 55 probability 1`
   2. `tc filter add dev tcont1 ingress flower skip_sw indev gem3 classid 8000:3 action ok`

   GEM4:
   1. `tc qdisc replace dev tcont1 parent 8000:4 handle 1004 red limit 100k min 30k max 80k avpkt 1k burst 55 probability 1`
   2. `tc filter add dev tcont1 ingress flower skip_sw indev gem4 classid 8000:4 action ok`

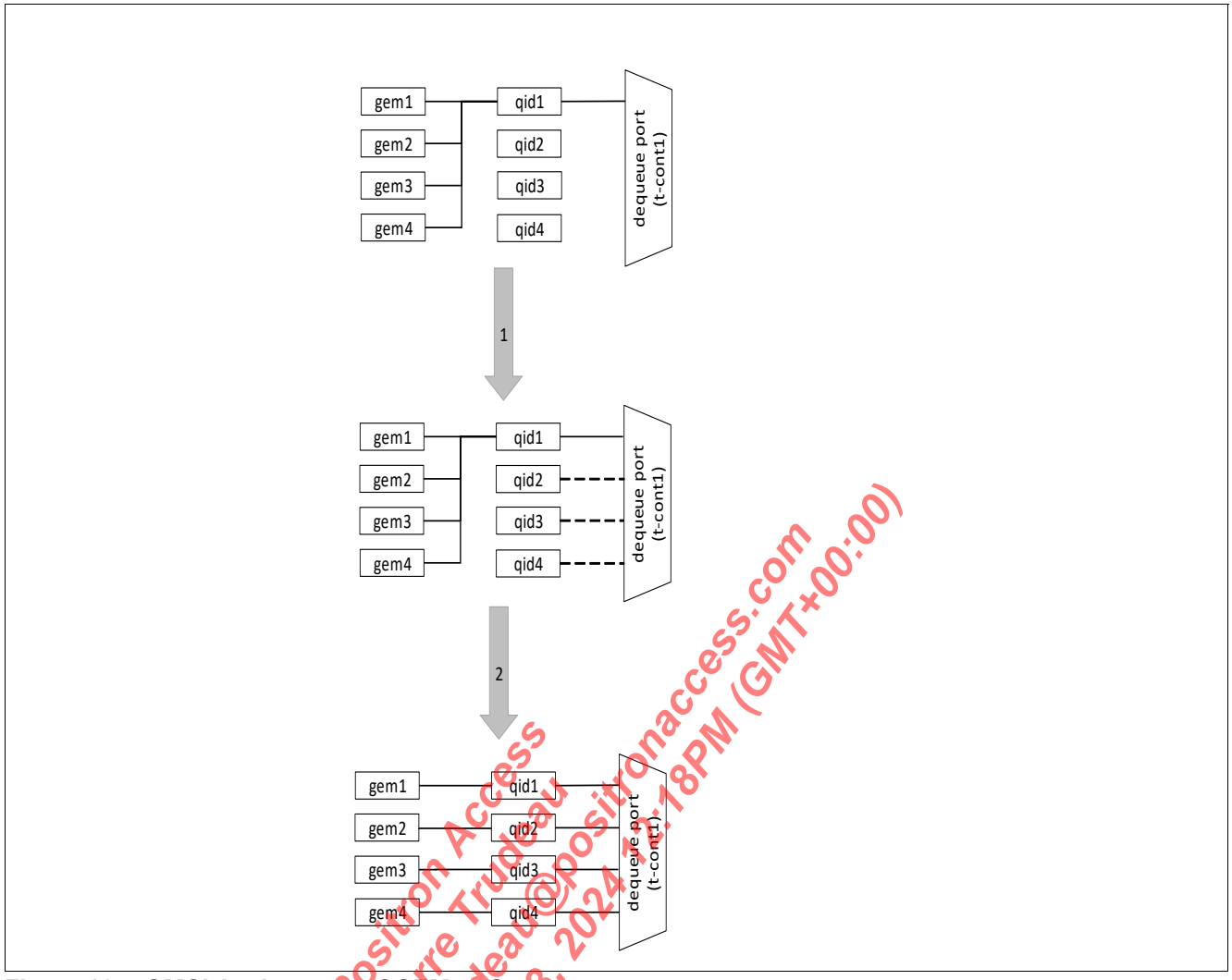


Figure 16 OMCI Assignment of GEM to Queue

Using tc-drr to Configure Weighted Round Robin (WRR) Single Stage Scheduling

The `tc-drr` does not automatically create child classes. It only creates the root qdisc and the classes must be created afterwards. **Figure 17** shows an example configuration. Each drr class is assigned a special quantum value proportional to each queue weight. The quantum values is calculated by multiplying the queue weight by the internal quanta value of the QoS engine transmit manager, for example:

quantum = weight * quanta = 2 * 4K.

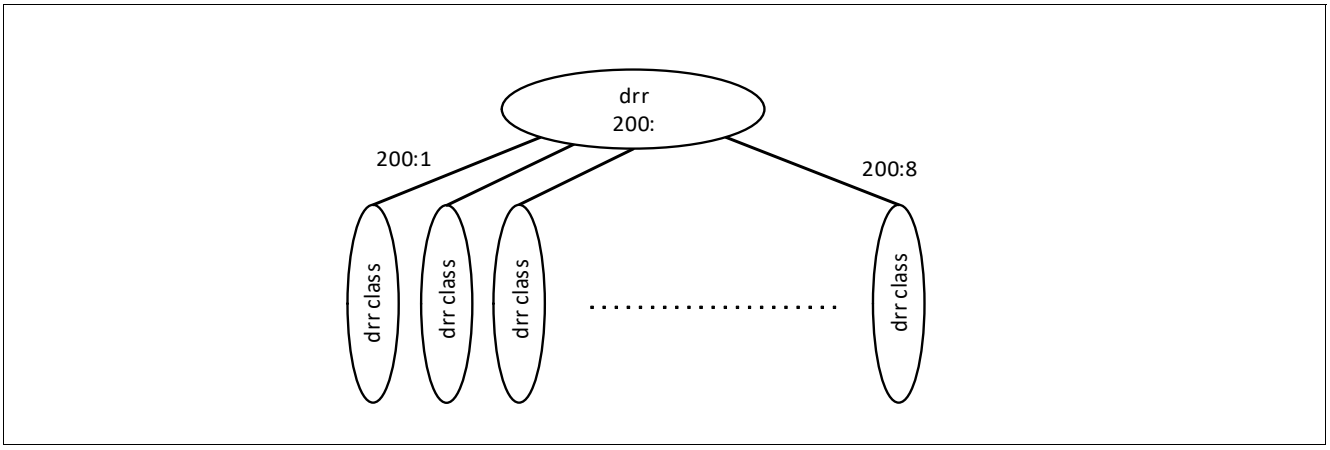


Figure 17 Linux tc drr Configuration

Figure 17 shows an example of command sequence to configure WRR on T-CONT. The last step is similar to the SP configuration where the assignment of the GEM ports to the queues happen. The example uses equal weight for each queue.

```
tc qdisc add dev tcont200 root handle 200: drr

# Create 8 classes - one for each queue
tc class add dev tcont200 parent 200: classid 200:1 drr quantum 4096k
tc class add dev tcont200 parent 200: classid 200:2 drr quantum 4096k
tc class add dev tcont200 parent 200: classid 200:3 drr quantum 4096k
tc class add dev tcont200 parent 200: classid 200:4 drr quantum 4096k
tc class add dev tcont200 parent 200: classid 200:5 drr quantum 4096k
tc class add dev tcont200 parent 200: classid 200:6 drr quantum 4096k
tc class add dev tcont200 parent 200: classid 200:7 drr quantum 4096k
tc class add dev tcont200 parent 200: classid 200:8 drr quantum 4096k

# Create 8 queues - one for each GEM port
tc filter add dev tcont200 ingress priority 1 protocol all flower indev gem301
skip_sw classid 200:1 action ok
tc filter add dev tcont200 ingress priority 2 protocol all flower indev gem302
skip_sw classid 200:2 action ok
tc filter add dev tcont200 ingress priority 3 protocol all flower indev gem303
skip_sw classid 200:3 action ok
tc filter add dev tcont200 ingress priority 4 protocol all flower indev gem304
skip_sw classid 200:4 action ok
tc filter add dev tcont200 ingress priority 5 protocol all flower indev gem305
skip_sw classid 200:5 action ok
tc filter add dev tcont200 ingress priority 6 protocol all flower indev gem306
skip_sw classid 200:6 action ok
tc filter add dev tcont200 ingress priority 7 protocol all flower indev gem307
skip_sw classid 200:7 action ok
tc filter add dev tcont200 ingress priority 8 protocol all flower indev gem308
skip_sw classid 200:8 action ok
```


Strict priority (SP) and Weighted Round Robin (WRR) Mixed Mode Scheduling

The mixed mode is configured by using these commands. Depending on the configuration, the SP and WRR queues number must be changed accordingly.

```
# SP+WRR
# 4 SP queues at priority 1-4 and 4 WRR Queues at priority 5
# Commands for addition
# Create MQPRIO Qdisc
tc qdisc add dev eth0_0 root handle 202: mqprio num_tc 8 map 0 1 2 3 4 5 6 7   queues
1@0 1@1 1@2 1@3 1@4 1@5 1@6 1@7 hw 2
tc qdisc add dev eth0_0 handle ffff: clsact

# create 4 SP queues
tc filter add dev eth0_0 ingress flower skip_sw classid 202:1 action ok
tc filter add dev eth0_0 ingress flower skip_sw classid 202:2 action ok
tc filter add dev eth0_0 ingress flower skip_sw classid 202:3 action ok
tc filter add dev eth0_0 ingress flower skip_sw classid 202:4 action ok

# create WRR Qdisc and attach it to child 5 of MQPRIO Qdisc

tc qdisc add dev eth0_0 parent 202:5 handle 9000: drr

tc class add dev eth0_0 parent 9000: classid 9000:1 drr quantum 8192
tc class add dev eth0_0 parent 9000: classid 9000:2 drr quantum 16384
tc class add dev eth0_0 parent 9000: classid 9000:3 drr quantum 32768
tc class add dev eth0_0 parent 9000: classid 9000:4 drr quantum 65536

tc filter add dev eth0_0 ingress flower skip_sw classid 9000:1 action ok
tc filter add dev eth0_0 ingress flower skip_sw classid 9000:2 action ok
tc filter add dev eth0_0 ingress flower skip_sw classid 9000:3 action ok
tc filter add dev eth0_0 ingress flower skip_sw classid 9000:4 action ok

# commands for deletion
# we delete from pref higher to lower (delete WRR queues then SP Queues)
tc filter del dev eth0_0 ingress pref 49152
tc filter del dev eth0_0 ingress pref 49151
tc filter del dev eth0_0 ingress pref 49150
tc filter del dev eth0_0 ingress pref 49149
tc filter del dev eth0_0 ingress pref 49148
tc filter del dev eth0_0 ingress pref 49147
tc filter del dev eth0_0 ingress pref 49146
tc filter del dev eth0_0 ingress pref 49145

# delete WRR Qdisc and MQPRIO Qdisc
…
```

3.7.2 Downstream QoS

The downstream QoS is configured on individual downstream LAN ports. The queues are added for each LAN port. Each LAN port is visualized as scheduler to which queues are attached with assigned priorities or weights.

- It is implemented by using Linux tc mqprio for strict priority scheduling functions.
- Linux tc does not offer WRR or WFQ scheduling. The tc requires extension to perform WRR or WFQ configuration.
- The QoS HAL (hardware abstraction layer) offers a `ndo_tc_setup()` callback function such as `pp_qos_setup()` which must be initialized to the LAN-side network device corresponding `ndo_tc_setup()` function.
- The TBF of tc is used to create traffic shapers.

The configuration of the downstream QoS is the same as the setup described in **Upstream QoS**. **Figure 18** shows the software module top level architecture.

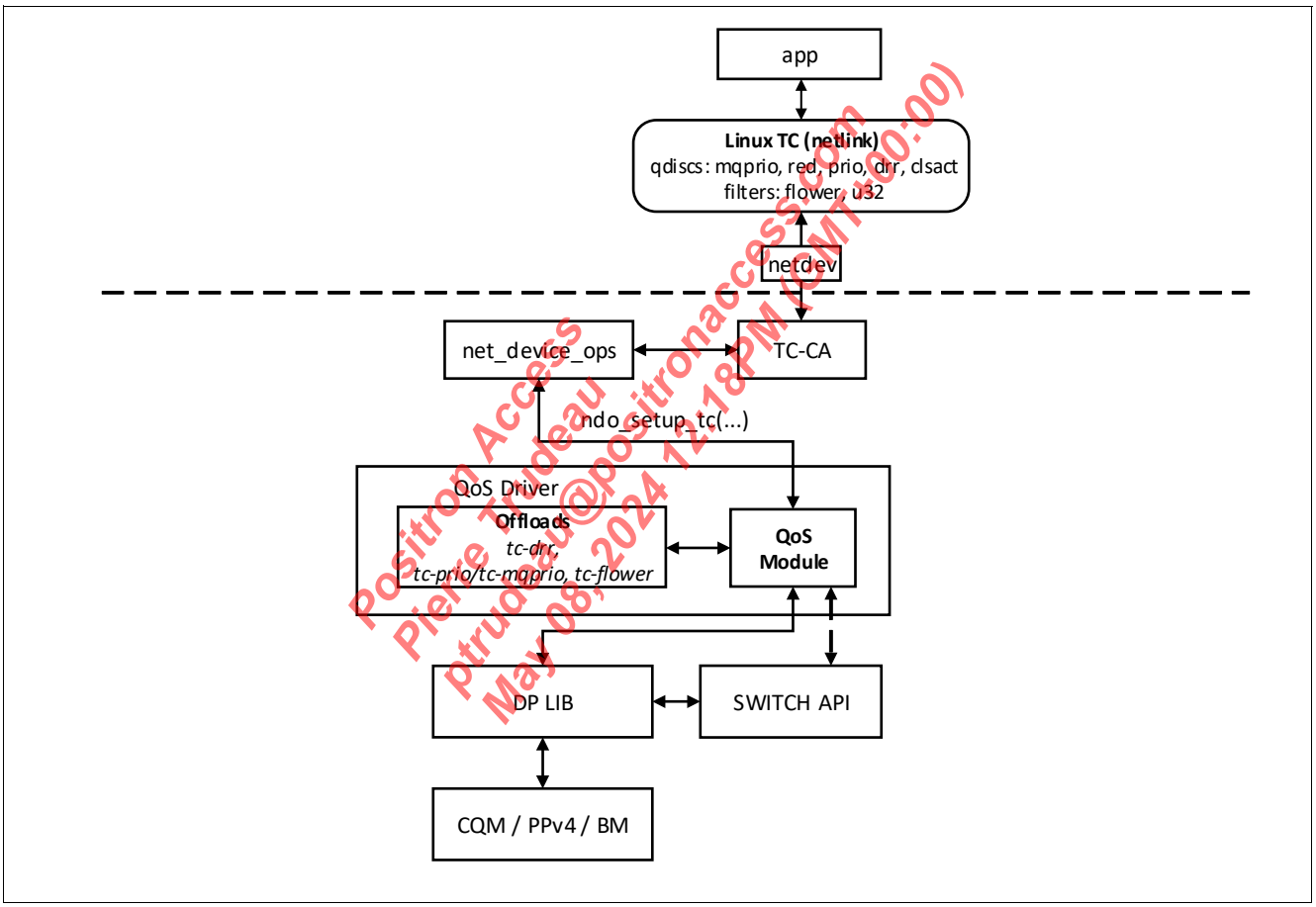


Figure 18 Software Modules Involved in Linux TC-CA qdisc Offloading

When a user space application creates and configures a qdisc and corresponding filters for classifying traffic to specific qdisc, the netlink message goes to the traffic control classifier action (TC-CA) subsystem in the Linux kernel. The corresponding qdisc/filter module checks and parses the parameters passed by the user space application. When the network driver that created the network device on which the qdisc/filter is registered supports hardware offload, then the driver `net_device_ops` callback `ndo_setup_tc` is called. This hook is implemented by the QoS driver and when it is called, it applies the necessary hardware configuration by calling the necessary APIs from the lower layer drivers.

The common operations implemented by a Linux qdisc are defined in `struct Qdisc_ops`.

Figure 19 shows this structure in details. The configuration parameters are offloaded in the qdisc `init` hook.

These are some examples:

```
int omci_qdisc_init(struct Qdisc *sch, struct nlattr *arg)
{
        struct net_device *dev = qdisc_dev(sch);
[…]
        dev->netdev_ops->ndo_setup_tc(dev, TC_SETUP_OMCI, &omciopt);
[…]
}
```

The `ndo_setup_tc` calls the required data path APIs to configure the hardware with the specified parameters. **Figure 20** shows the standard qdiscs used to build the hierarchical setup.

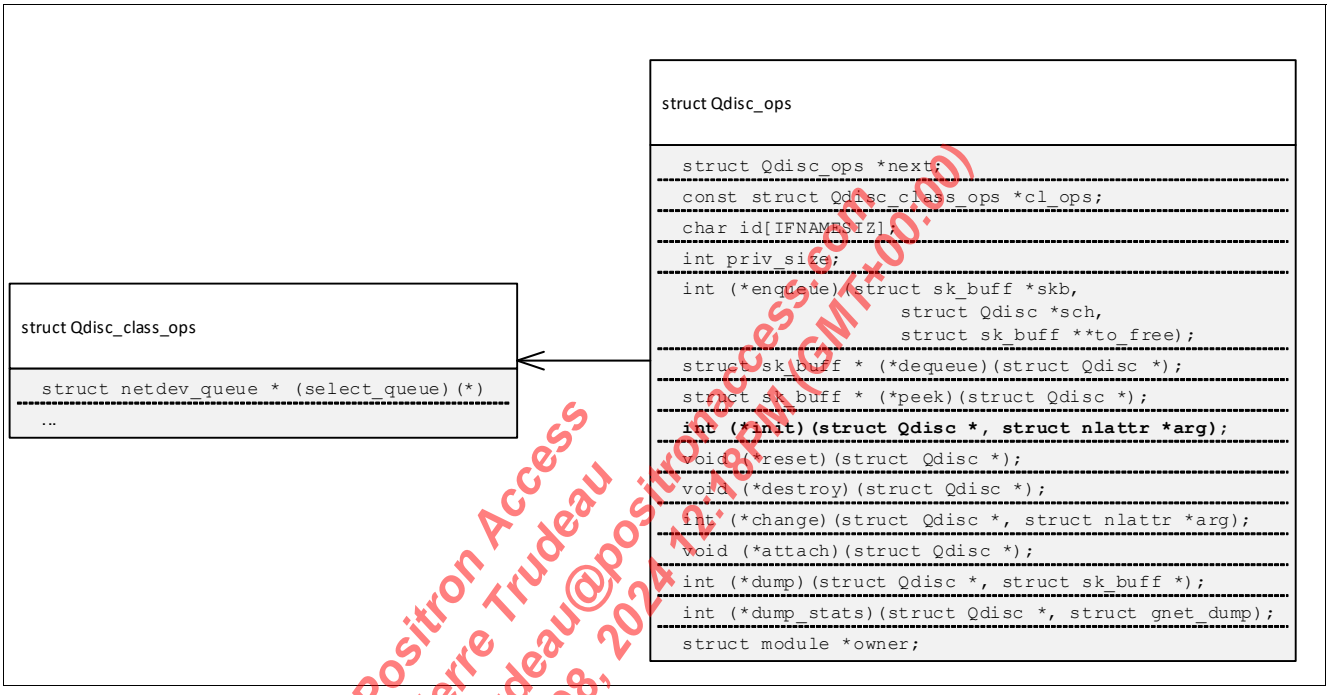


Figure 19 Linux qdisc Operation Hooks

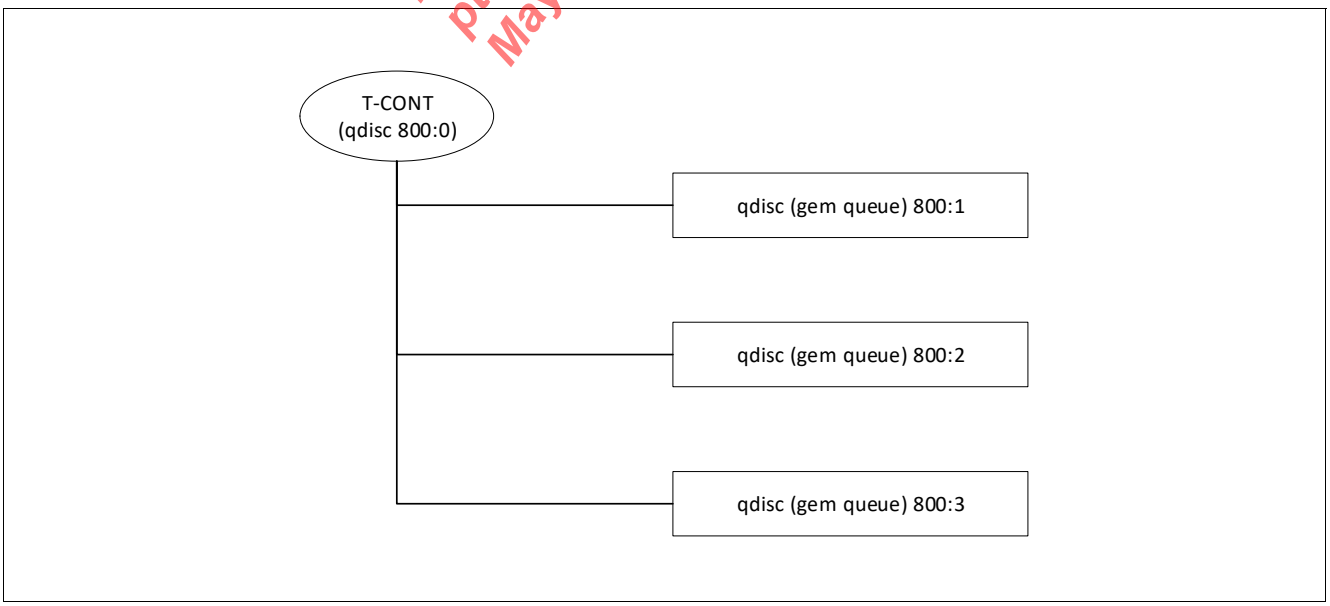


Figure 20 Example of OMCI qdisc Hierarchy

The QoS configuration implementations must use as much as possible the currently available kernel qdiscs. Whenever possible, only the offloading feature must be added. **Figure 21** shows an example of qdisc tree created with available qdisc. The figure shows shaper and policers that are described in the subsequent chapters.

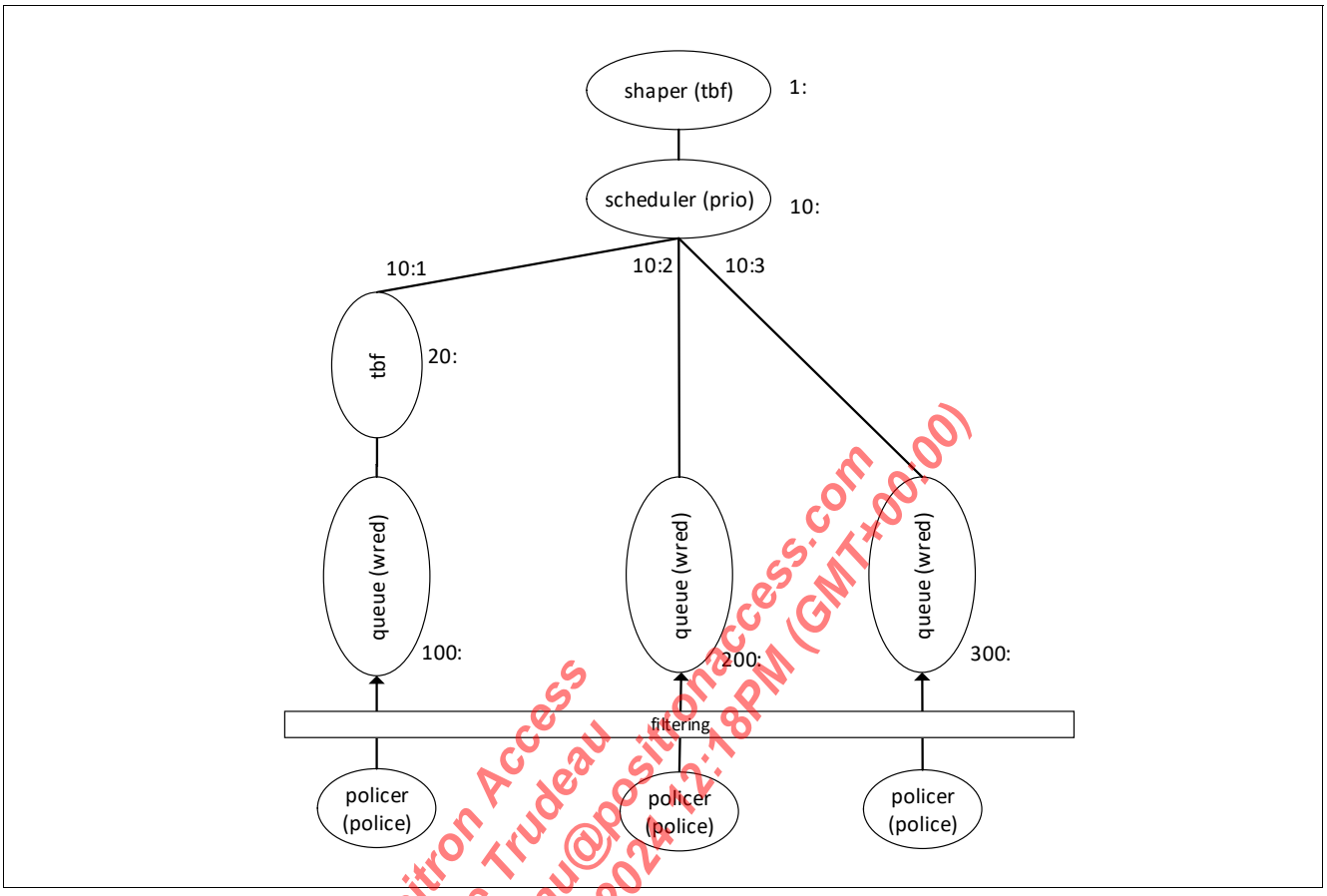


Figure 21 Tree Offload Example

The policers described in **Figure 21** are not part of the qdisc tree as they are implemented as filter qdiscs in Linux. In **Figure 21**, they are depicted to show that traffic is policed or colored before entering the respective queue.

The downstream packets are assigned to the corresponding queue based on the traffic class assigned during the packet processing in hardware. There are two cases supported for assigning downstream flows: one based directly when using the *tc* command on GEM port to queue mapping and another based on pbit to queue:

- GEM to queue assignment command for downstream
  ```
  tc filter add dev eth0_0 ingress flower skip_sw indev gem1024 classid 1:1 action ok
  ```
- Pbit to queue assignment command for downstream
  ```
  tc filter add dev eth0_0 ingress flower skip_sw vlan_prio classid 1:1 action ok
  ```

Figure 22 shows the stages of traffic class assignment in hardware.

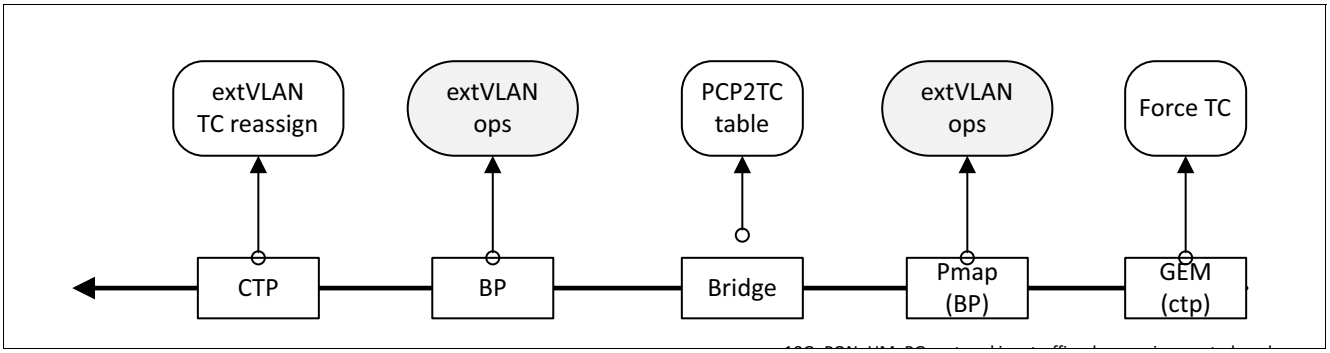


Figure 22 Downstream Traffic Class Assignment

In case of GEM to queue assignment, the traffic class is assigned on the CTP port. The CTP port is configured in force traffic class mode and the traffic class configured takes precedence over all other traffic class assignments.

In case of IEEE 802.1p (pbit) to queue mapping, the traffic class is assigned in two stages: one before the VLAN translation and another after the VLAN translation. First, the switch pcp (pbit) to traffic class table is consulted and the packet traffic class is set. After that, the VLAN operation is applied and in case the pbit is changed, the addition extended VLAN operations on the CTP port reassign the traffic class to the new value according to the pbit change. The traffic class reassignment operations are mapping pbit to traffic class according IEEE 802.1p.

WRED Configuration

The WRED offloading is implemented by using the Linux tc red qdisc. This qdisc has support for hardware offloading in the Linux kernel. As the WRED qdisc requires min, max drop thresholds and max drop probability for green and yellow packets, two red qdiscs are used. The parent is used for the green thresholds setup and the second one for the yellow packets (see **Figure 23**).

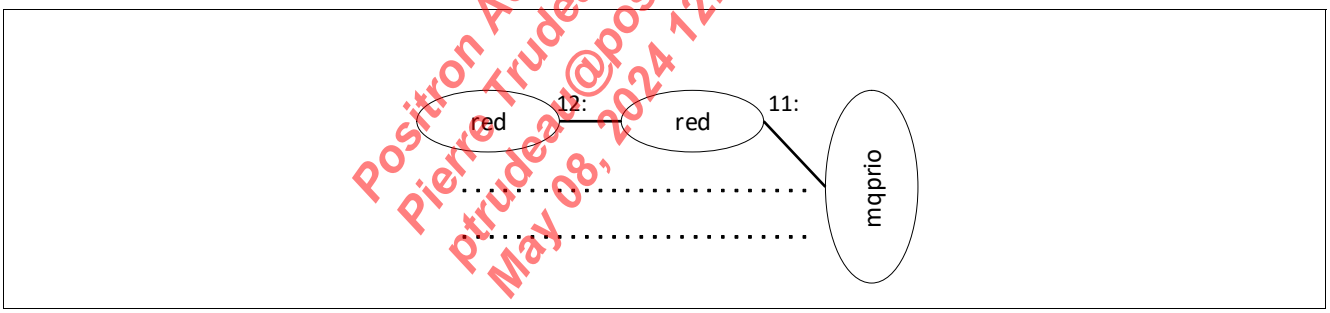


Figure 23 Mapping WRED to tc red qdiscs

The WRED queue qdisc seen in **Figure 21** is replaceable by two red qdiscs, a fact that the driver implementing the offloading callback (`ndo_setup_tc`) must take into consideration. The offloading parameters for the tc red qdisc already contain a reference to its parent qdisc handle. The callback checks whether it has another red qdisc as a parent by using the function `qdisc_lookup` and the parent ID in the `struct tc_red_qopt_offload`. When it is the first qdisc, the parameters for the green packets must be cached. When it is the second red qdisc, the previously saved parameters for the green packets are offloaded to the hardware, together with the new yellow parameters coming with the second qdisc offloading configuration structure. **Figure 23** shows the graphical representation of this setup. See this example of a red qdisc setup:

```
tc qdisc add dev eth0_0 root handle 11: red limit 100k min 30k max 80k avpkt 1k burst
55 probability 0.02
tc qdisc add dev eth0_0 parent 11: handle 12: red limit 100k min 30k max 80k avpkt
1k burst 55 probability 0.02
```

3.7.2.1 Disable Downstream QoS

When the UNI side interface data rate is always higher than the downstream data rate received from the OLT, the downstream QoS handling can be disabled (this is called “shortcut mode). It saves some power dissipation and processing latency because the data traffic does not require to pass the external DRAM memory. It also saves data bandwidth on the DRAM memory interface.

When the QoS is disabled, downstream packet scheduling according to packet priority and packet discarding according to WRED are not done. All packets are forwarded in the sequence of their reception. When the downstream interface is slower than the arriving packet rate and the queues fill up, packets are tail-dropped regardless of their priority.

Static Configuration of Downstream QoS Bypass Mode

The configuration is done at compile time in the device tree (DTS).

The default setting provided in `/dts/prx300.dtsi` to enable the QoS is:

```
/* 0 - 10G full QoS */
/* 1 - 10G short cut */
gsw_mode = <0>;
```

To disable downstream QoS and enable the shortcut mode, change the setting to:

```
gsw_mode = <1>;
```

3.7.3 Meters and Color Marking

The meters (policers) are used to measure the data rate of dedicated data streams and mark them.

Figure 24 shows a simplified data flow model for the PON OMCI usage of traffic policers (meters). Meters are only applied on ingress of bridge ports or termination points in the OMCI context.

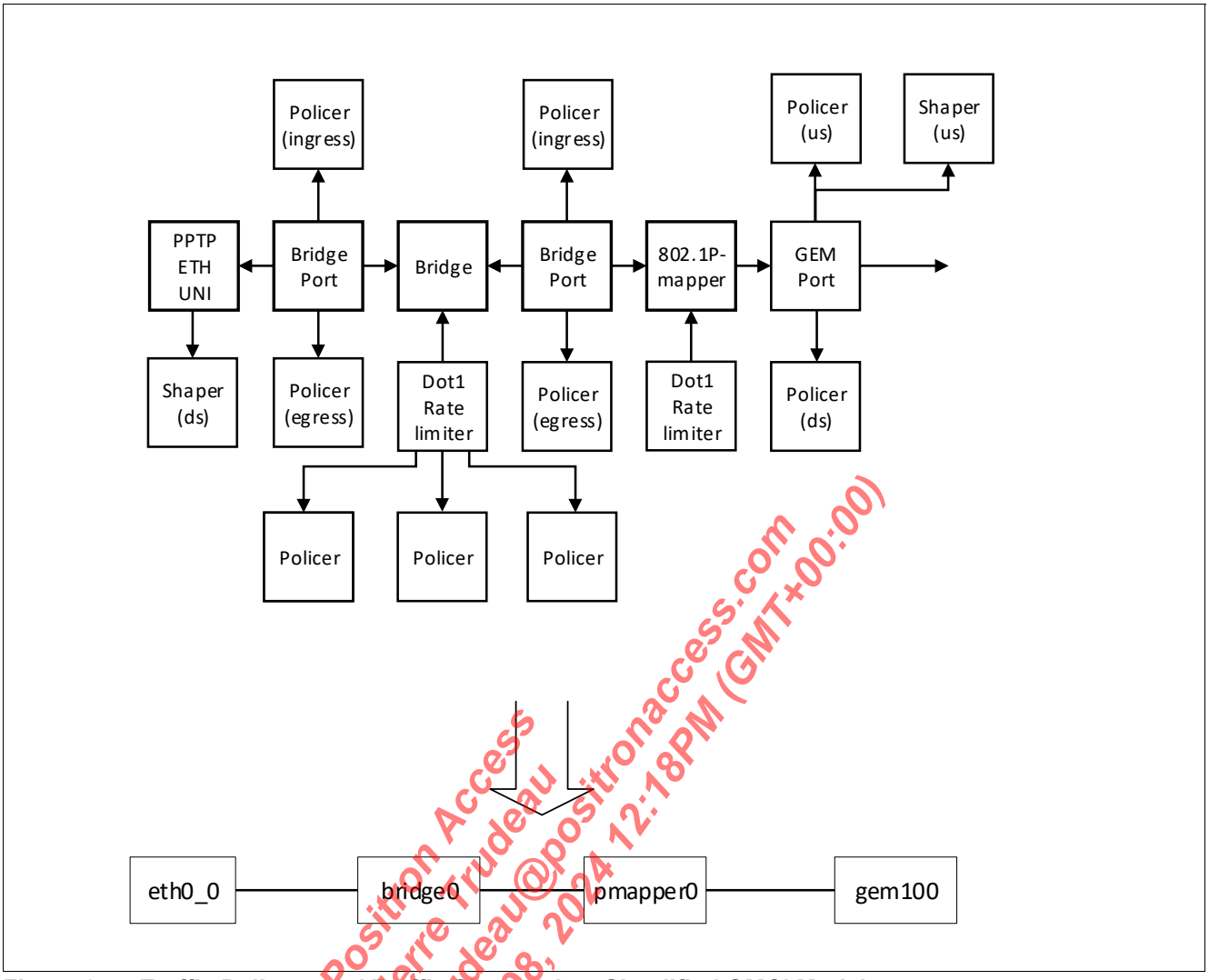


Figure 24 Traffic Policers and Traffic Shapers in a Simplified OMCI Model

Figure 25 depicts the software components involved in the policer configuration.

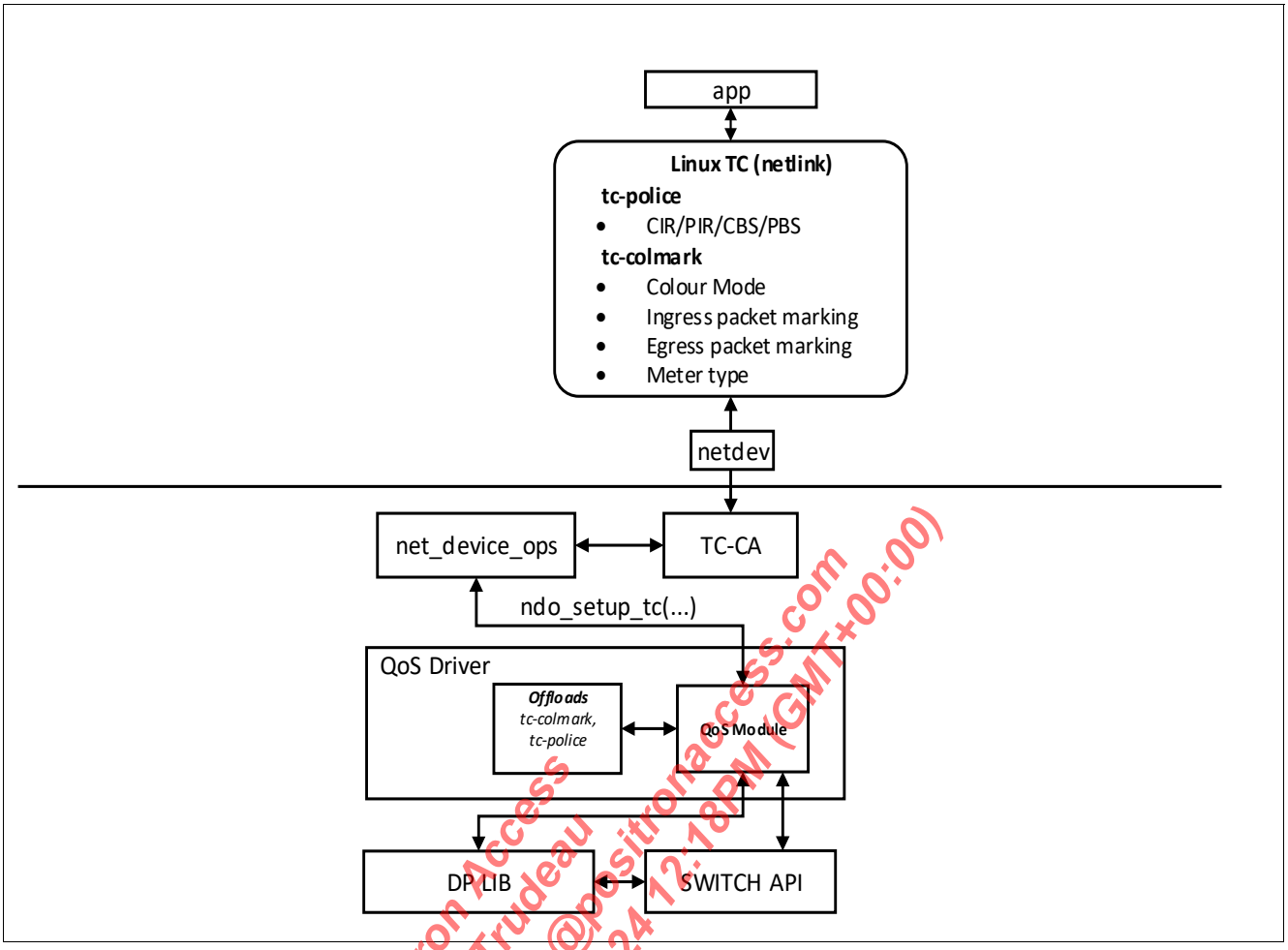


Figure 25 Software Modules Involved in Traffic Policer Configuration

The configuration is triggered by the user space using the tc netlink interface for the police action. These parameters must be configured for a meter/policer:

- CIR - Committed Information Rate [bytes/s]
- PIR - Peak Information Rate [bytes/s]
- CBS - Committed Burst Size [bytes]
- PBS - Peak Burst Size [bytes]
- Color mode - color-blind or color-aware
- Color marking for ingress/egress
  - No marking
  - Internal marking
  - DEI
  - PCP_8P0D
  - PCP_7P1D
  - PCP_6P2D
  - PCP_5P3D
  - DSCP_AF
- Meter type - srTCM (single rate Three Color Marking) or trTCM (two rate Three Color Marking)

These are meter parameters mapping to the tc-police and tc-colmark action options:

- CIR - tc-police rate
- PIR - tc-police peakrate
- CBS - tc-police burst

- PBS - tc-police mtu/minburst
- Color mode - tc-colmark blind
- Color marking for ingress/egress - tc-colmark marker
- Meter type - tc-colmark mtype

Refer to **[46]**, **[47]**, and **[48]** for details.

These are example configuration commands for metering on the GEM CTP or bridge ports:

```
# BP meter
tc filter add dev eth0_0 egress pref 1111 protocol all flower skip_sw indev eth0_0
action police rate 10Mbit peakrate 50Mbit burst 30K mtu 40K pipe colmark mtype srTCM
# Delete
tc filter del dev eth0_0 egress pref 1111

# BP meter
tc filter add dev eth0_0 egress pref 1111 protocol all flower skip_sw indev eth0_0
action police rate 500Kbit peakrate 1Mbit burst 30K mtu 40K pipe colmark mtype trTCM

# Delete
tc filter del dev eth0_0 egress pref 1111

# CTP meter
tc filter add dev eth0_0 ingress pref 1111 protocol all flower skip_sw action police
rate 10Mbit peakrate 50Mbit burst 30K mtu 40K pipe colmark mtype srTCM
# Delete
tc filter del dev eth0_0 ingress pref 1111

# Pmapper(BP) meters
tc filter add dev pmapper4354 egress pref 1111 protocol all flower skip_sw action
police rate 10Mbit peakrate 16Mbit burst 30K mtu 40K pipe colmark mtype trTCM
tc filter add dev pmapper4354 ingress pref 1111 protocol all flower skip_sw action
police rate 2Mbit peakrate 50Mbit burst 30K mtu 40K pipe colmark mtype srTCM
# Delete
tc filter del dev pmapper4354 egress pref 1111
tc filter del dev pmapper4354 ingress pref 1111

# GEM port(CTP) meter
tc filter add dev gem1023 egress pref 1111 protocol all flower skip_sw action police
rate 10Mbit peakrate 10Mbit burst 20K mtu 30K pipe colmark mtype trTCM
# Delete meter
tc filter del dev gem1023 egress pref 1111

# GEM port(CTP) meter
tc filter add dev gem1024 ingress pref 1111 protocol all flower skip_sw action police
rate 10Mbit peakrate 100Mbit burst 20K mtu 30K
# Delete meter
tc filter del dev gem1024 ingress pref 1111
```

Traffic Shapers

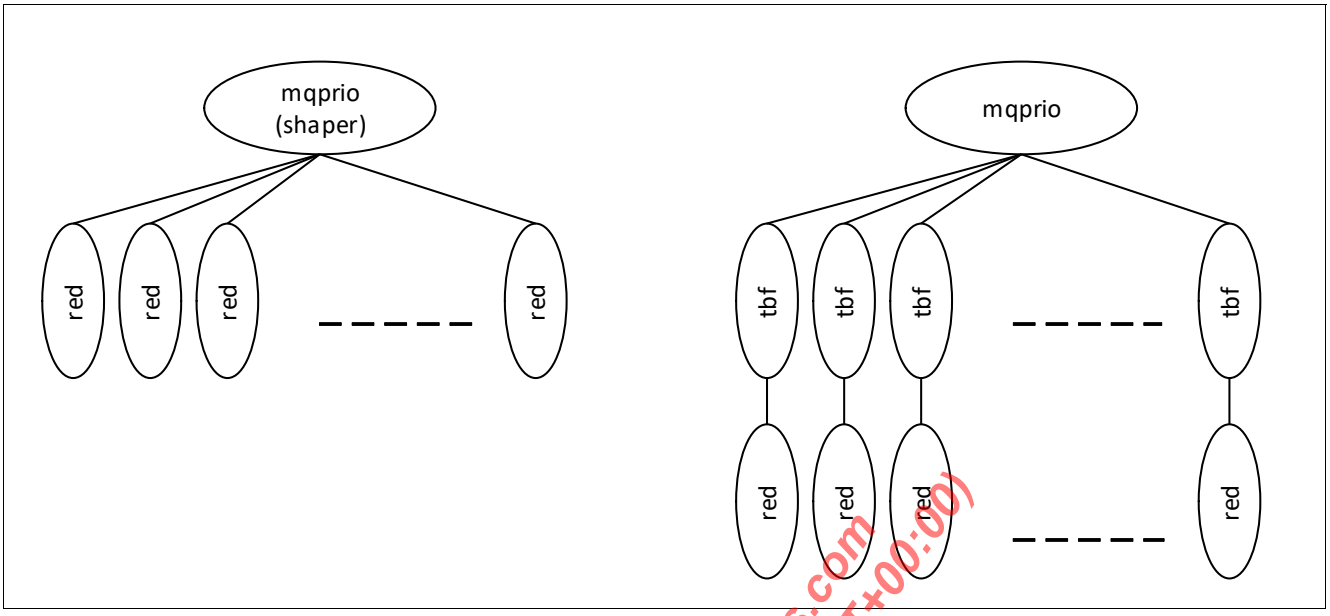


Figure 26 Possible Linux TC-CA qdiscs for Traffic Shaper Configuration

Figure 26 depicts possible ways to configure shapers in Linux using the TC-CA qdisc. The tc-mqprio is able to set up shapers for each of its child queues. It is possible to insert the tc-tbf on the other side in the qdisc tree between the different qdiscs. It implements a classical token bucket shaper. For example, use these commands to setup shapers using tc-tbf. The parent ID must point to the parent qdisc leaf node/queue. When using a tc-prio qdisc with handle 102 and when shapers to the first two queues must be added, the parent IDs 102:1 and 102:2 must be used.

```
# single rate shaper CIR = "rate", CBS = "burst"
tc qdisc add dev eth0_0 parent 102:1 handle 333: tbf rate 2Mbit burst 10k limit 100
# dual rate shaper PIR = "peakrate", PBS = "minburst"
tc qdisc add dev eth0_0 parent 102:2 handle 433: tbf rate 2Mbit burst 10k peakrate
10Mbit minburst 20K limit 100

# tc qdisc show dev tcont32768

" qdisc mqprio 8000: root
qdisc red 2010: parent 1010:1 limit 250Kb min 225Kb max 250Kb ecn:false
harddrop:false adaptive:false
qdisc red 2014: parent 1014:1 limit 250Kb min 225Kb max 250Kb ecn:false
harddrop:false adaptive:false
qdisc red 2012: parent 1012:1 limit 250Kb min 225Kb max 250Kb ecn:false
harddrop:false adaptive:false
qdisc red 1011: parent 4000:2 limit 500Kb min 450Kb max 500Kb ecn:false
harddrop:false adaptive:false
qdisc red 2016: parent 1016:1 limit 250Kb min 225Kb max 250Kb ecn:false
harddrop:false adaptive:false
qdisc red 1013: parent 4000:4 limit 500Kb min 450Kb max 500Kb ecn:false
harddrop:false adaptive:false
qdisc red 1015: parent 4000:6 limit 500Kb min 450Kb max 500Kb ecn:false
harddrop:false adaptive:false
```

```
qdisc red 1017: parent 4000:8 limit 500Kb min 450Kb max 500Kb ecn:false
harddrop:false adaptive:false
qdisc red 2011: parent 1011:1 limit 250Kb min 225Kb max 250Kb ecn:false
harddrop:false adaptive:false
qdisc drr 4000: parent 8000:1
qdisc red 2013: parent 1013:1 limit 250Kb min 225Kb max 250Kb ecn:false
harddrop:false adaptive:false
qdisc red 1010: parent 4000:1 limit 500Kb min 450Kb max 500Kb ecn:false
harddrop:false adaptive:false
qdisc red 2015: parent 1015:1 limit 250Kb min 225Kb max 250Kb ecn:false
harddrop:false adaptive:false
qdisc red 1014: parent 4000:5 limit 500Kb min 450Kb max 500Kb ecn:false
harddrop:false adaptive:false
qdisc red 1012: parent 4000:3 limit 500Kb min 450Kb max 500Kb ecn:false
harddrop:false adaptive:false
qdisc red 2017: parent 1017:1 limit 250Kb min 225Kb max 250Kb ecn:false
harddrop:false adaptive:false
qdisc red 1016: parent 4000:7 limit 500Kb min 450Kb max 500Kb ecn:false
harddrop:false adaptive:false
qdisc clsact ffff: parent ffff:fff1 "

# Add shaper in above qdisc tree to queue (classid 4000:1, red qdisc 2010:1)
tc qdisc add dev tcont32768 parent 2010:1 handle 8888: tbf rate 8Mbit burst 16k
peakrate 10Mbit minburst 20K limit 100
```

3.8 Multicast Packet Forwarding

The PON ONU has special GEM channels meant for carrying multicast or broadcast traffic only in the downstream direction. They are different from bidirectional GEM ports as they are shared amongst multiple ONUs. Here are the main points for multicast handling.

- Multicast GEMs are mapped to network devices similar to the unicast GEMs.
- OMCI creates one or more multicast GEM ports in the downstream direction for multicast data packets.
- The multicast bridges are not explicitly shown in Linux rather they are programmed directly in GSWIP by MCC.
- When a multicast GEM or broadcast GEM is attached to more than one bridge then the special multicast UNI network device must be used to enable multicast traffic sharing between the bridges. The naming of these devices uses the postfix “mcc”, for example `eth0_0_mcc`. **Figure 27** shows an example with two bridges. In this case, the multicast UNI network device is added to the first bridge (*BR0*). This enables multicast traffic sharing on both UNI interfaces. Further control of the multicast flows coming to these interfaces is then later applied by the OMCI configuration.
- The multicast data packets do not come to Linux but get replicated by GSWIP in the offload path itself.
- The upstream multicast data packets are never sent in the upstream direction, which is enforced by mapping the multicast GEM to the drop queue.

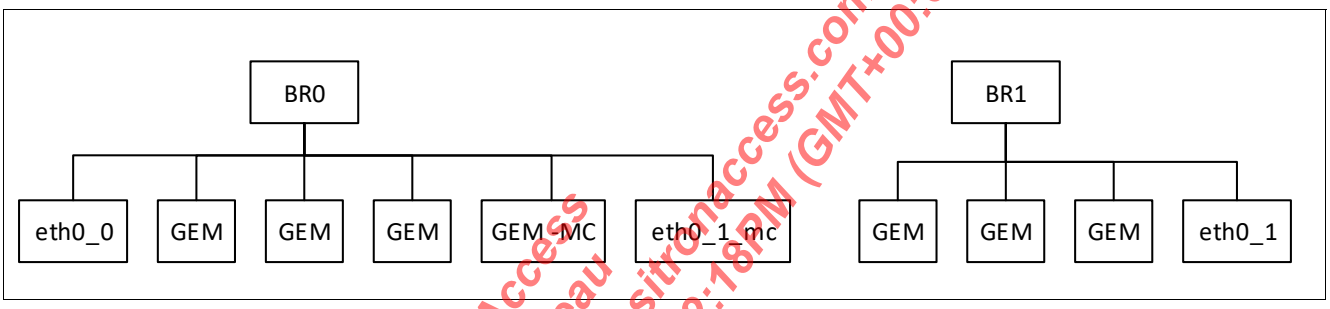


Figure 27 Multicast Traffic Handling for Two UNI Ports in Separate Bridges

Figure 28 shows an example of OMCI multicast traffic flow, together with the corresponding mapping to Linux network devices. When the snooping configuration is enabled, all multicast control packets are received in Linux. The mcc daemon uses Linux raw sockets for this purpose.

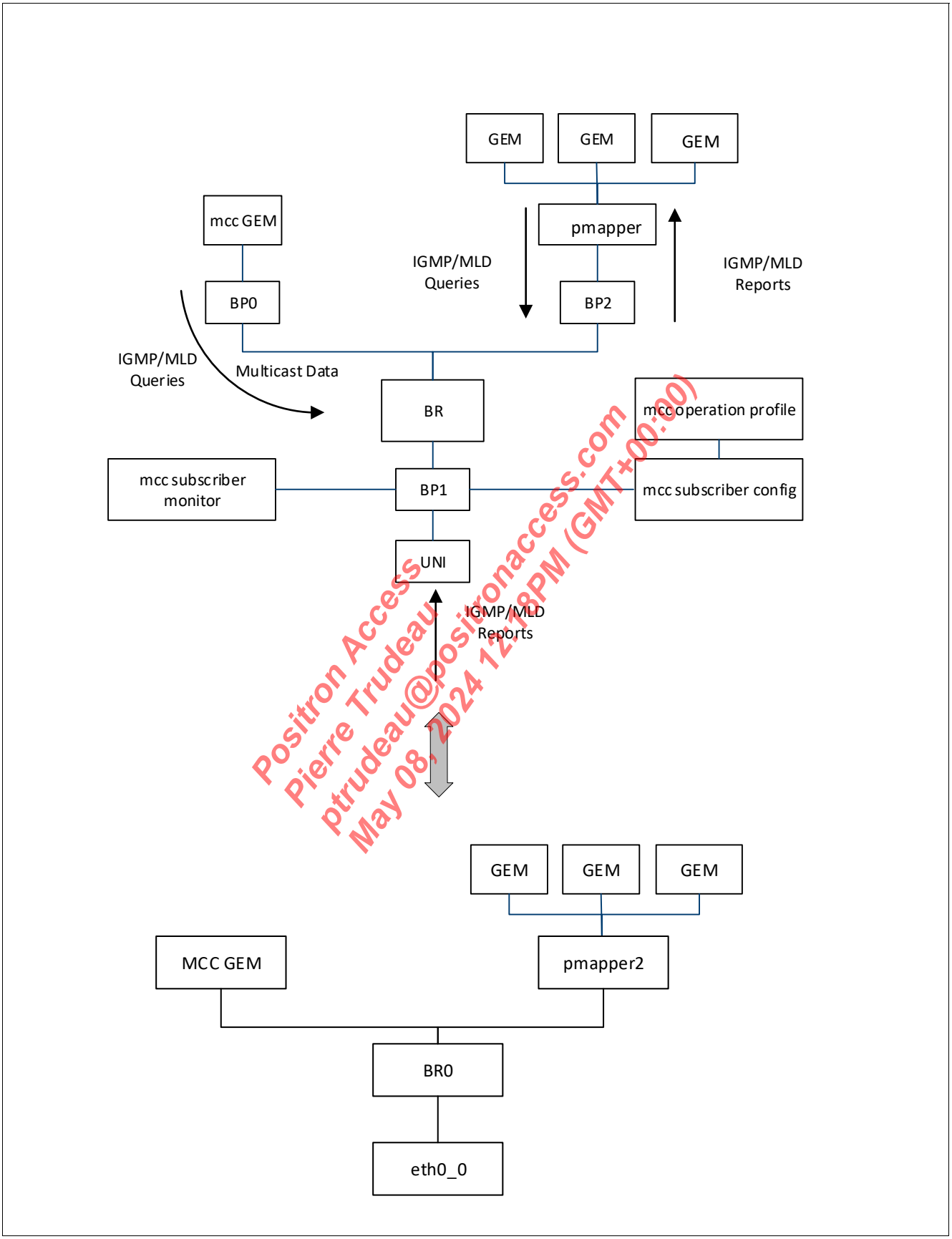


Figure 28 Multicast Bridges and Control Packets Path

3.8.1 Multicast Protocol Handling

The multicast control packets (IGMP and MLD) are received on the Linux raw socket and handled by the `mcc` application to configure the multicast forwarding in the switch.

The `mcc` application is reused from the legacy FALC ON project with some modification. The main differences are:

- Switch forwarding table programming
  - Instead of calling the `onu` driver for direct firmware/hardware table programming, the switch API is used to program the L3 multicast table.
  - This is done for the pre-defined well-known addresses as well as for the multicast flows added through the IGMP/MLD protocol.
  - Multicast switch API functions, namely `GSW_MULTICAST_TABLE_ENTRY_ADD`, `GSW_MULTICAST_TABLE_ENTRY_REMOVE` and `GSW_MULTICAST_TABLE_ENTRY_READ`.
- VLAN handling
  - The VLAN handling is no longer done automatically by the firmware/hardware but must be done within the `mcc` software for the IGMP/MLD control packets.
  - The VLAN handling is defined by OMCI and must be adapted. Instead of configuring the Extended VLAN table, the software must add/modify/remove VLANs on the IGMP/MLD control packets.
  - VLAN handling of the IGMP/MLD packets is different per direction (upstream/downstream), as defined by OMCI (multicast operations profile).
  - VLAN handling for the multicast data packets must be configured into the switch (downstream only).
- Format of received/transmitted IGMP/MLD packets
  - The packet format must be adapted, as the device uses dedicated meta data on a raw network interface.
- IGMP rate limitation
  - The maximum rate of upstream IGMP/MLD packets must be limited. This requires adaptation as the existing implementation also includes a hardware policer protecting the software from IGMP/MLD-based DoS attacks.

Check this to identify the necessary changes:

- The MCC application must invoke the MCC Kernel helper module to perform configuration in the hardware switch module (GSWIP).
  Add or remove the layer 3 multicast table of the switch.
- The OMCI performs dynamic control on MCC application. It is important that the OMCI daemon is able to write to the MCC daemon configuration file such as `mcc.conf` and trigger the MCC application through a signal, such as SIGHUP.

PON Multicast Driver

Figure 29 depicts the PON multicast driver top level architecture. The PON multicast driver provides user space and kernel space interfaces for configuring the multicast L3 forwarding, IGMP/MLD snooping and multicast bridging. In the kernel space, the DP manager APIs for VLAN translation are used to configure the necessary multicast VLAN operations defined by OMCI whenever necessary. In the user space, these operations are configured using the tc command by defining multicast flows based on the MAC addresses.

IPv4 multicast

```
tc filter add dev eth0_0 ingress pref 100 protocol 802.1q flower dst_mac
01:00:5E:00:00:00/25 skip_sw action vlan pop
```

IPv6 multicast

```
tc filter add dev eth0_0 ingress pref 101 protocol 802.1q flower dst_mac
33:33:00:00:00:00/16 skip_sw action vlan pop
```

IPv4 multicast

```
tc filter add dev eth0_0 ingress pref 100 protocol 802.1q flower dst_mac
01:00:5E:00:00:00/25 skip_sw action vlan push id 100
```

IPv6 multicast

```
tc filter add dev eth0_0 ingress pref 101 protocol 802.1q flower dst_mac
33:33:00:00:00:00/16 skip_sw action vlan push id 100
```

IPv4 multicast

```
tc filter add dev eth0_0 ingress pref 100 protocol 802.1q flower dst_mac
01:00:5E:00:00:00/25 skip_sw action vlan modify id 100
```

IPv6 multicast

```
tc filter add dev eth0_0 ingress pref 101 protocol 802.1q flower dst_mac
33:33:00:00:00:00/16 skip_sw action vlan modify id 100
```

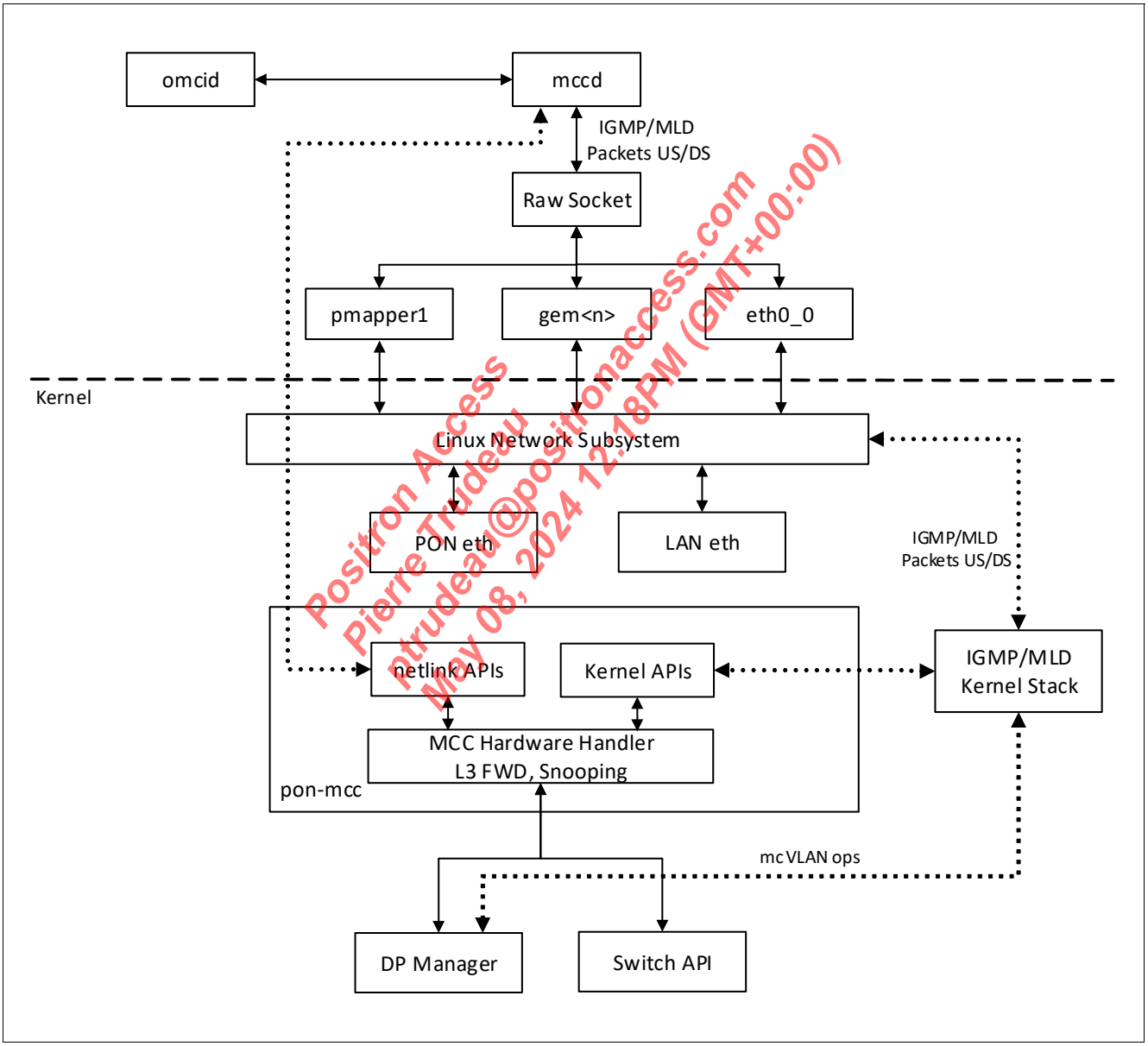


Figure 29 PON Multicast Driver Software Architecture

The handling of the multicast IGMP/MLD control packets is done in software, either in user space or kernel space. The software takes care of the correct VLAN translation and modification of the control packets.

3.9 Broadcast Packet Forwarding

For the broadcast handling of PON ONU, there are special unidirectional downstream only GEMs which carry broadcast packets from the OLT to the ONU. These GEMs are not allowed to sent broadcast packets on the upstream direction. The handling of the broadcast GEMs in multiple bridge mode is exactly the same as that of the multicast GEM but the `eth0_<n>_bc` interfaces must be used instead.

3.10 Packet Reinsertion

The network packets are reinserted back by the software to the hardware for further processing. For example, this is required when the packets must go through the VLAN processing in upstream or downstream. A use case is the handling of the multicast IGMP/MLD control packets. They are received by the software module handling the multicast unmodified and the module reinserts these packets through the hardware processing.

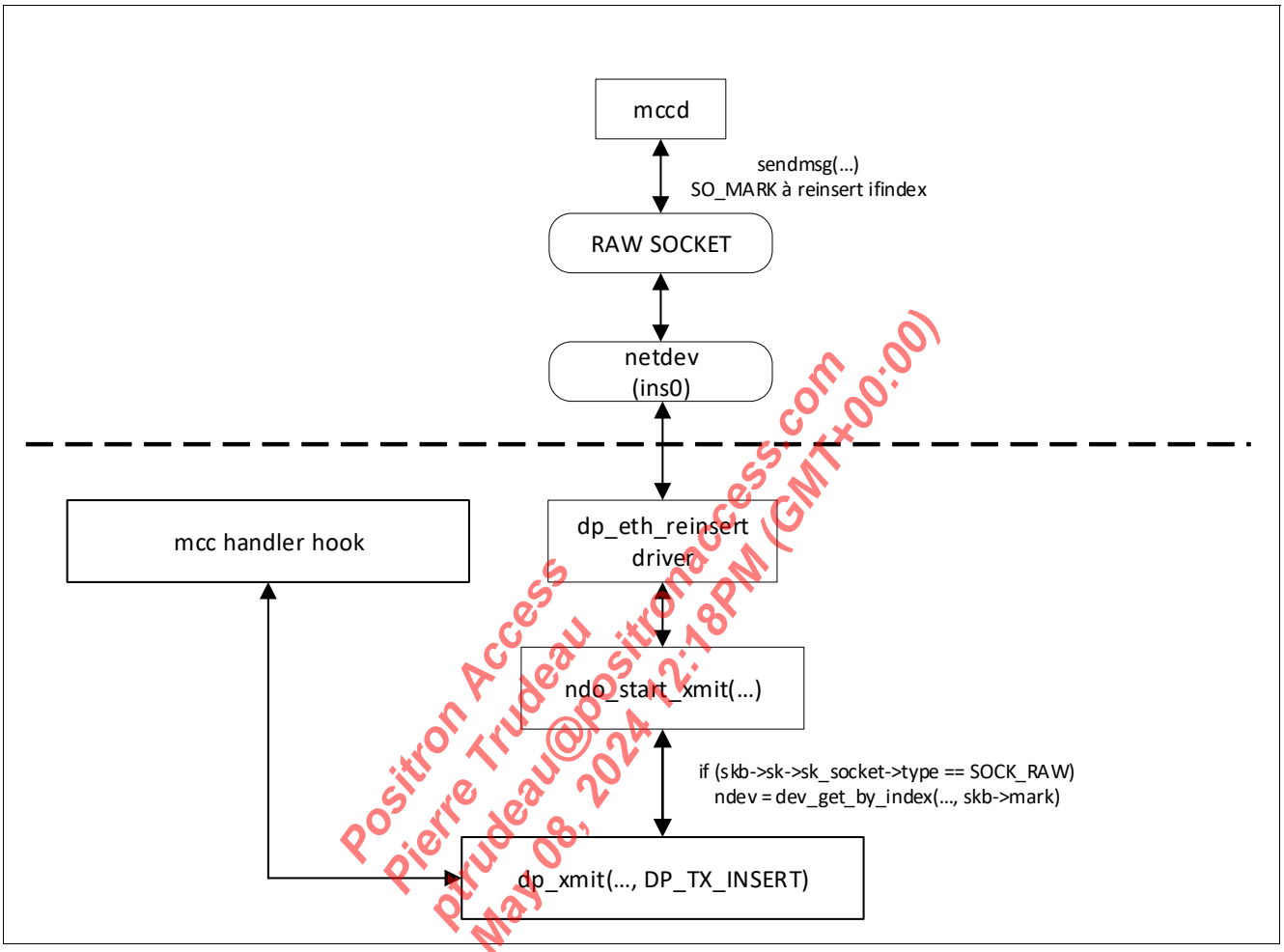


Figure 30 Packet Re-insertion in Case of Multicast

In **Section 3.10**, the packet reinsertion flows for multicast control packets are shown in the case of insertion from the user space and kernel space. The kernel space module handling the IGMP/MLD packets inserts packets back by calling `dp_xmit()`, provided by the DP manager kernel module. The reinsertion is specified by the special flag `DP_TX_INSERT`. The reinsertion from the user space involves another driver which provides a special network device for packet insertion. The user space application uses a Linux socket send function as `sendmsg`. The application must specify the insertion point where the network device must insert the packet in the `SO_MARK` socket option. Based on the network device specified in the `SO_MARK`, the `dp_eth_reinsert` driver adds in turn the necessary hardware port information in the skb buffer and calls the `dp_xmit` function.

3.11 Flow-based VLAN Aware Forwarding

The QoS driver supports flow-based VLAN forwarding using the `tc-mirred` action. The tc defined flows are offloaded to the switch flow table which takes precedence when the bridge port learning is disabled. The flow-based forwarding is supported only for single UNI/LAN port. The forwarding rules in hardware are always assigned to the CTP port belonging to the UNI port. The driver allocates a block of 64 entries for each CTP.

There are entries in this block reserved for special forwarding rules. **Figure 31** displays the layout.

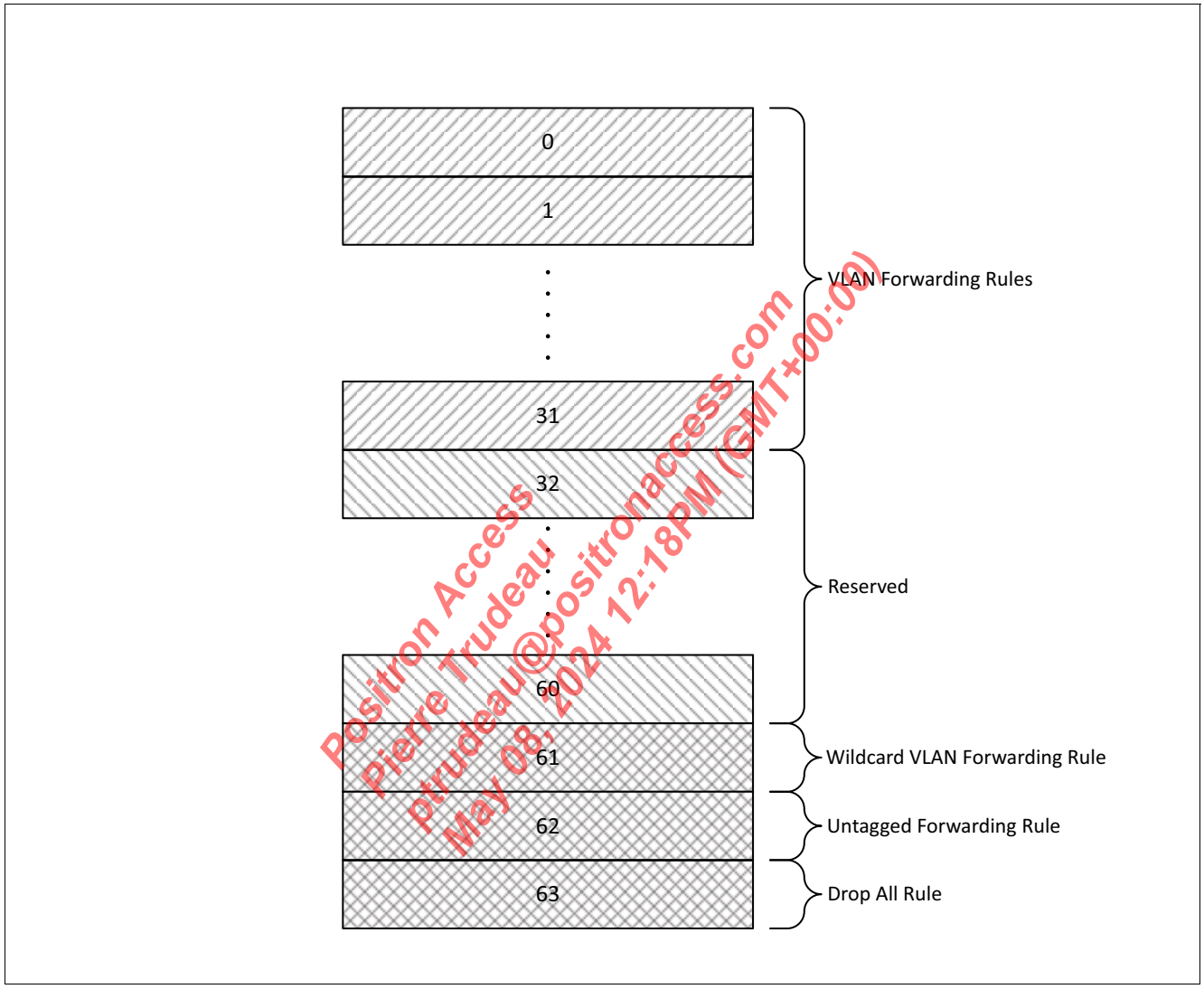


Figure 31 Flow Block Used for VLAN Forwarding on LAN Port

Use these commands to configure the individual flows. The comments show the assigned index in the table, assuming the issuing order did not change.

```
# forward VID=507 ingressing on eth0_0 to pmapper4354 (flow block index = 0)
tc filter add dev eth0_0 ingress pref 501 protocol 802.1q flower skip_sw vlan_id 507
action mirred egress redirect dev pmapper4354

# forward VID=508 ingressing on eth0_0 to pmapper4355 (flow block index = 1)
tc filter add dev eth0_0 ingress pref 502 protocol 802.1q flower skip_sw vlan_id 508
action mirred egress redirect dev pmapper4355

# forward VID=509 ingressing on eth0_0 to pmapper4356 (flow block index = 2)
```

```
tc filter add dev eth0_0 ingress pref 503 protocol 802.1q flower skip_sw vlan_id 509
action mirred egress redirect dev pmapper4356

# forward VID=* (all VIDs) ingressing on eth0_0 to pmapper4356 (flow block index = 61)
tc filter add dev eth0_0 ingress pref 504 protocol 802.1q flower skip_sw action
mirred egress redirect dev pmapper4356

# forward all untagged packets ingressing on eth0_0 to pmapper4356 (flow block index
= 62)
tc filter add dev eth0_0 ingress pref 505 protocol all flower skip_sw action mirred
egress redirect dev pmapper4356

# drop all packets ingressing on eth0_0 (flow block index = 63)
tc filter add dev eth0_0 ingress pref 506 protocol all flower skip_sw action mirred
egress redirect dev sw257 drop
```

Note: The TPID value in these commands is not considered in the offload. This means that specifying 802.1ad configures the same rules in the hardware.

3.12 Ethernet Driver

LAN (UNI) Port Speed Control

The Ethernet driver is responsible for the LAN port speed configuration. It provides an ethtool compatible interface used to set the corresponding speed modes. **Figure 32** shows an example of configuration flow triggered by an OMCI daemon when the UNI port speed is changed.

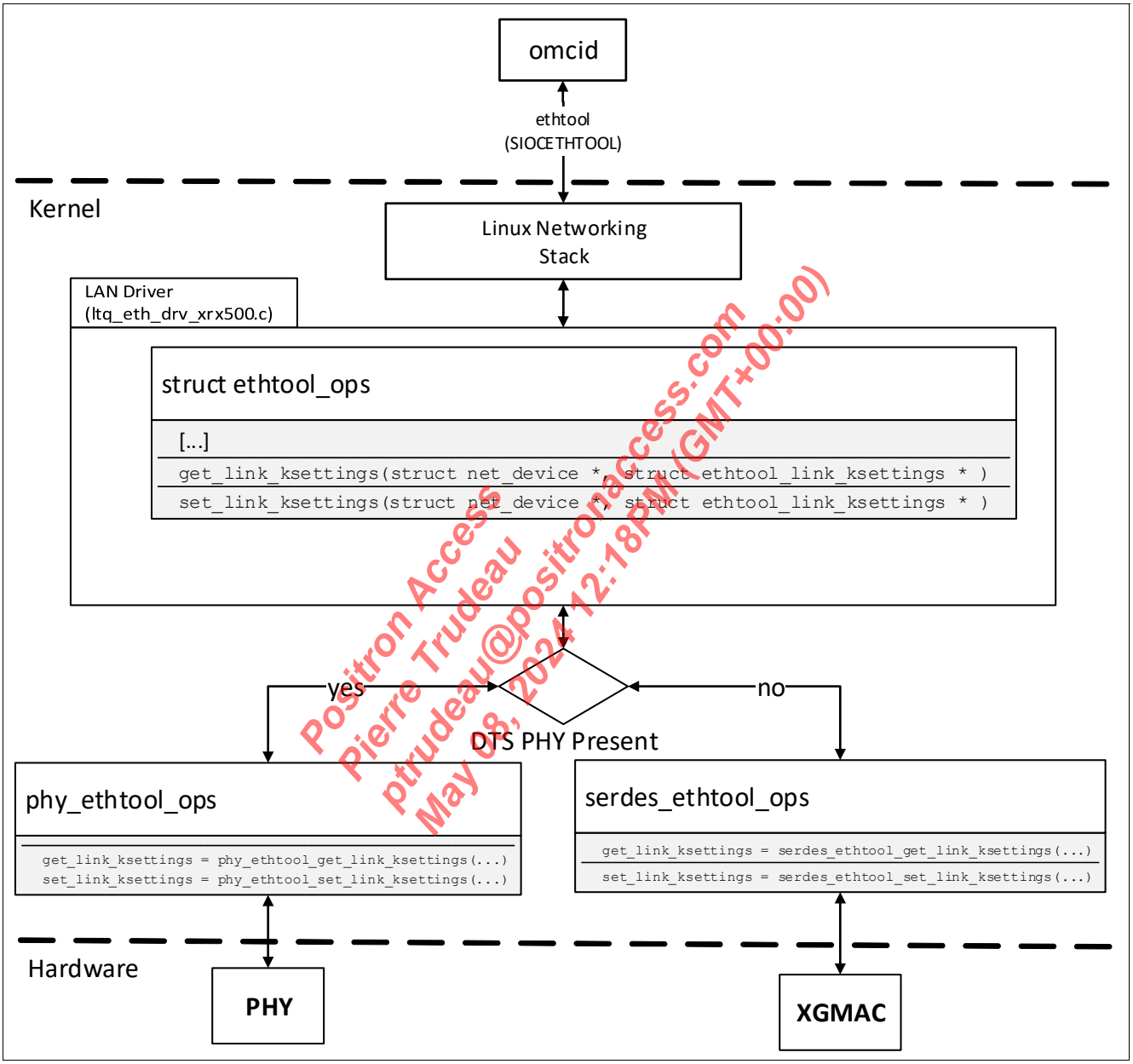


Figure 32 Ethernet Driver Configuration Flow

The driver implements two sets of the `ethtool_ops` operation structure. One is used when an external PHY is connected to the MAC and the other when no PHY is connected, plain SerDes mode. In SerDes mode, these speed settings are available for configuration:

- 1G: 1000baseKX_Full
- 2.5G: 2500baseX_Full
- 10G: 10000baseKR_Full

Note: The ethtool is used to show the current speed, duplex mode and supported link modes.

3.13 Packet Parsing

The GSWIP hardware accelerator parses the incoming packet headers to identify the layer 2 packet type and position of the layer 3/layer 4 headers.

The dedicated packet types are detected and stored in parser flags used to further classify the packet and apply the desired handling. Table 18 lists the parser flag numbers and their usage. It is possible to set multiple flags at the same time, depending on the packet properties. Up to 64 parser flags are supported by the hardware but only a subset is currently in use.

Table 18 Parser Flag Definitions

Parser Flag No.	Packet Type
0	No flag is assigned.
1	Parser processing end. This flag is inverted in the flow classification engine when flags are used as key and carried to MPE.
2	Reserved
3	Reserved
4	The packet is length encapsulated.
5	Reserved
6	Reserved
7	Reserved
8	The internal special tag is present.
9	There is at least one VLAN tag.
10	There are at least two VLAN tags.
11	There are at least three VLAN tags.
12	There are at least four VLAN tags.
13	The packet is SNAP encapsulated.
14	The packet is a PPPoE session packet.
15	Packet has at least one IP header. The first IP header is IPv4.
16	Packet has at least one IP header. The first IP header is IPv6.
17	Packet has at least two IP headers. The second IP header is IPv4.
18	Packet has at least two IP headers. The second IP header is IPv6.
19	There are more than three VLAN tags present.
20	There is a TCP header present.
21	UDP is sent over the first IP header.
22	There is an IGMP header present[1)].
23	Reserved
24	The first IPv6 next header is DES, HOP, or ROUT.
25	The packet is a TCP acknowledge (ACK) packet.
26	Reserved
27	The packet is an EAPOL packet.
28	The second IPv6 next header is DES, HOP, or ROUT.
29	Reserved
30	Reserved

Table 18 Parser Flag Definitions (cont'd)

Parser Flag No.	Packet Type
31	Reserved
32	Reserved
33	Reserved
34	Reserved
35	Reserved
36	This is an IPv4 multicast packet.
37	The Ethertype is 0x888E.
38	Reserved
39	Reserved
40	Reserved
41	Reserved
42	Reserved
43	Reserved
44	Reserved
45	Reserved
46	Reserved
47	Reserved
48	This is a DHCPv4 packet.
49	This is a DHCPv6 packet.
50	Reserved
51	Reserved
52	Reserved
53	Reserved
54	Reserved
55	Reserved
56	Reserved
57	Reserved
58	Reserved
59	Reserved
60	Reserved
61	Reserved
62	Reserved
63	Reserved

1) The IGMP flag is NOT triggered by the MLD packets. The MLD packets must be filtered individually.

3.14 Packet Prioritization to the CPU

To enable QoS for the data traffic targeted at the software, multiple (8) queues are provided. The traffic type per queue must be configured as rules in the PCE filter table. **Table 19** shows the prioritization set by default.

Table 19 CPU Traffic Prioritization

Traffic Class	Traffic Type
8 (highest)	OMCI management packets
9	PTP (IEEE 1588) management packets
10	CFM (Ethernet OAM) packets/IP host
11	Packets addressed to the LCT MAC address
	IGMP/MLD packets
12	ICMP and ICMPv6 packets
	Multicast MAC address for IPv6 Neighbor Discovery Protocol (NDP)
13	ARP and ARP response packets
14	Reserved for future use
15 (lowest)	All other packets

For software revisions up to 10G PON Chipset System Package 1.16.x:

The traffic class mappings are defined in `pon_net_lib/src/pon_net_netlink.c`.

For software revisions starting from 10G PON Chipset System Package 1.17.0:

The traffic class mappings are defined in `pon_net_lib/src/pon_net_init.c`.

The shapers per queue are configured to avoid the software being overrun by excessive traffic, for example, as seen in DoS attacks or incorrectly configured networks. The prioritization ensures that high priority packets still reach the CPU even under traffic overload conditions.

3.15 Packet Rate Limitation to the CPU

To prevent DoS attacks to the CPU, the data rate targeted at the CPU is limited to 10 Mbit/s by default.

3.15.1 Configuration up to Release 1.16

The configuration is modifiable in `/etc/init.d/cpu-queues-init.sh`. The default configuration is:

```
#!/bin/sh /etc/rc.common

# run after "boot" script, which loads the drivers
START=11

# Configure static TC queues from the PON IP to the CPU to limit the traffic
# to 10MBit/s. This is used to prevent an overload of the CPU and a DoS based
# on this. The OMCI packets are getting a high priority to still reach the CPU.
# Each queue has red threshold to prevent buffer exhaustion under heavy traffic.
boot() {
        tc qdisc add dev pon0 root handle 1: prio bands 8 priomap 0 1 2 3 4 5 6 7
        tc qdisc add dev pon0 handle ffff: clsact

        # Highest priority queue doesn't have queue shaper as it is OMCI queue
       #tc qdisc add dev pon0 parent 1:1 handle 433: tbf rate 8Mbit burst 10k peakrate
10Mbit minburst 20K limit 100
        tc filter add dev pon0 ingress flower skip_sw classid 1:1 action ok cookie 08

       tc qdisc add dev pon0 parent 1:2 handle 434: tbf rate 8Mbit burst 10k peakrate
10Mbit minburst 20K limit 100
        tc qdisc add dev pon0 parent 434:1 handle 1001: red limit 115K min 16K max
16K avpkt 1k burst 120 probability 1.0
        tc filter add dev pon0 ingress flower skip_sw classid 1:2 action ok cookie 09

       tc qdisc add dev pon0 parent 1:3 handle 435: tbf rate 8Mbit burst 10k peakrate
10Mbit minburst 20K limit 100
        tc qdisc add dev pon0 parent 435:1 handle 1002: red limit 115K min 16K max
16K avpkt 1k burst 120 probability 1.0
        tc filter add dev pon0 ingress flower skip_sw classid 1:3 action ok cookie 0a

       tc qdisc add dev pon0 parent 1:4 handle 436: tbf rate 8Mbit burst 10k peakrate
10Mbit minburst 20K limit 100
        tc qdisc add dev pon0 parent 436:1 handle 1003: red limit 115K min 16K max
16K avpkt 1k burst 120 probability 1.0
        tc filter add dev pon0 ingress flower skip_sw classid 1:4 action ok cookie 0b

       tc qdisc add dev pon0 parent 1:5 handle 437: tbf rate 8Mbit burst 10k peakrate
10Mbit minburst 20K limit 100
        tc qdisc add dev pon0 parent 437:1 handle 1004: red limit 115K min 16K max
16K avpkt 1k burst 120 probability 1.0
        tc filter add dev pon0 ingress flower skip_sw classid 1:5 action ok cookie 0c

       tc qdisc add dev pon0 parent 1:6 handle 438: tbf rate 8Mbit burst 10k peakrate
10Mbit minburst 20K limit 100
```

```
        tc qdisc add dev pon0 parent 438:1 handle 1005: red limit 16K min 16K max
16K avpkt 1k burst 120 probability 1.0
       tc filter add dev pon0 ingress flower skip_sw classid 1:6 action ok cookie 0d

      tc qdisc add dev pon0 parent 1:7 handle 439: tbf rate 8Mbit burst 10k peakrate
10Mbit minburst 20K limit 100
        tc qdisc add dev pon0 parent 439:1 handle 1006: red limit 115K min 16K max
16K avpkt 1k burst 120 probability 1.0
       tc filter add dev pon0 ingress flower skip_sw classid 1:7 action ok cookie 0e

      tc qdisc add dev pon0 parent 1:8 handle 440: tbf rate 8Mbit burst 10k peakrate
10Mbit minburst 20K limit 100
        tc qdisc add dev pon0 parent 440:1 handle 1007: red limit 115K min 16K max
16K avpkt 1k burst 120 probability 1.0

        # Map the remaining TCs 0-7 and 15 to the low priority queue
       tc filter add dev pon0 ingress flower skip_sw classid 1:8 action ok cookie 00
       tc filter add dev pon0 ingress flower skip_sw classid 1:8 action ok cookie 01
       tc filter add dev pon0 ingress flower skip_sw classid 1:8 action ok cookie 02
       tc filter add dev pon0 ingress flower skip_sw classid 1:8 action ok cookie 03
       tc filter add dev pon0 ingress flower skip_sw classid 1:8 action ok cookie 04
       tc filter add dev pon0 ingress flower skip_sw classid 1:8 action ok cookie 05
       tc filter add dev pon0 ingress flower skip_sw classid 1:8 action ok cookie 06
       tc filter add dev pon0 ingress flower skip_sw classid 1:8 action ok cookie 07
       tc filter add dev pon0 ingress flower skip_sw classid 1:8 action ok cookie 0f
}
```

The OMCI packets traffic class is mapped using the `tc-trap` action on the `gem-omci` network device. The first nibble of the classid defines the traffic class, as shown in bold text.

```
# tc filter show dev gem-omci ingress
filter protocol all pref 65002 flower
filter protocol all pref 65002 flower handle 0x1 classid: fff8
  skip_sw
        action order 1: gact action trap
         random type none pass val 0
         index 17 ref 1 bind 1
```

3.15.2 Configuration from Release 1.17 Onwards

The maximum traffic rate that can be sent to the CPU is configured during the `pon_net_lib` initialization.

The source file `pon_net_init.c` provides these definitions by default:

```
#define CPU_QDISC_TBF_RATE 1000000
#define CPU_QDISC_TBF_PEAK_RATE 1250000
#define CPU_QDISC_TBF_BRUST_SIZE 10240
#define CPU_QDISC_TBF_PEAK_BRUST_SIZE 20480
#define CPU_QDISC_TBF_LIMIT 100

#define CPU_QDISC_RED_THRESHOLD 16384
#define CPU_QDISC_RED_PROBABILITY 1.0
#define CPU_QDISC_RED_LIMIT 117760
#define CPU_QDISC_RED_PARENT_QUEUE 1
```

It is possible to adapt these values as required.

3.16 IP Host Network Interface

Figure 33 shows how the VoIP application connects to the `iphost0` network interface if only a single instance is required. The MAC address is derived from the general base MAC address defined in the U-Boot environment.

If more than one instance of the IP Host interface are needed, then refer to the description in **Chapter 3.17**.

The IP address is either acquired through DHCP or configured by the OLT through OMCI.

The ingress and egress extended VLAN handling is performed by the software in the extended VLAN OMCI managed entity, based on tc commands (`pon_net_lib`). The VLAN priority is ignored and only the VLAN ID is filtered.

In the downstream direction, the VoIP data packets are received on a dedicated GEM port and detected based on MAC address and VLAN ID, using two dedicated PCE filter rules. The `pon_eth_drv` forwards the packets to the `iphost0` interface.

In the upstream direction, the `iphost_start_xmit` function of the `pon_eth_drv` inserts the packets in a dedicated `eth0_0_us` network interface which are then forwarded by the bridge to the IP host GEM port.

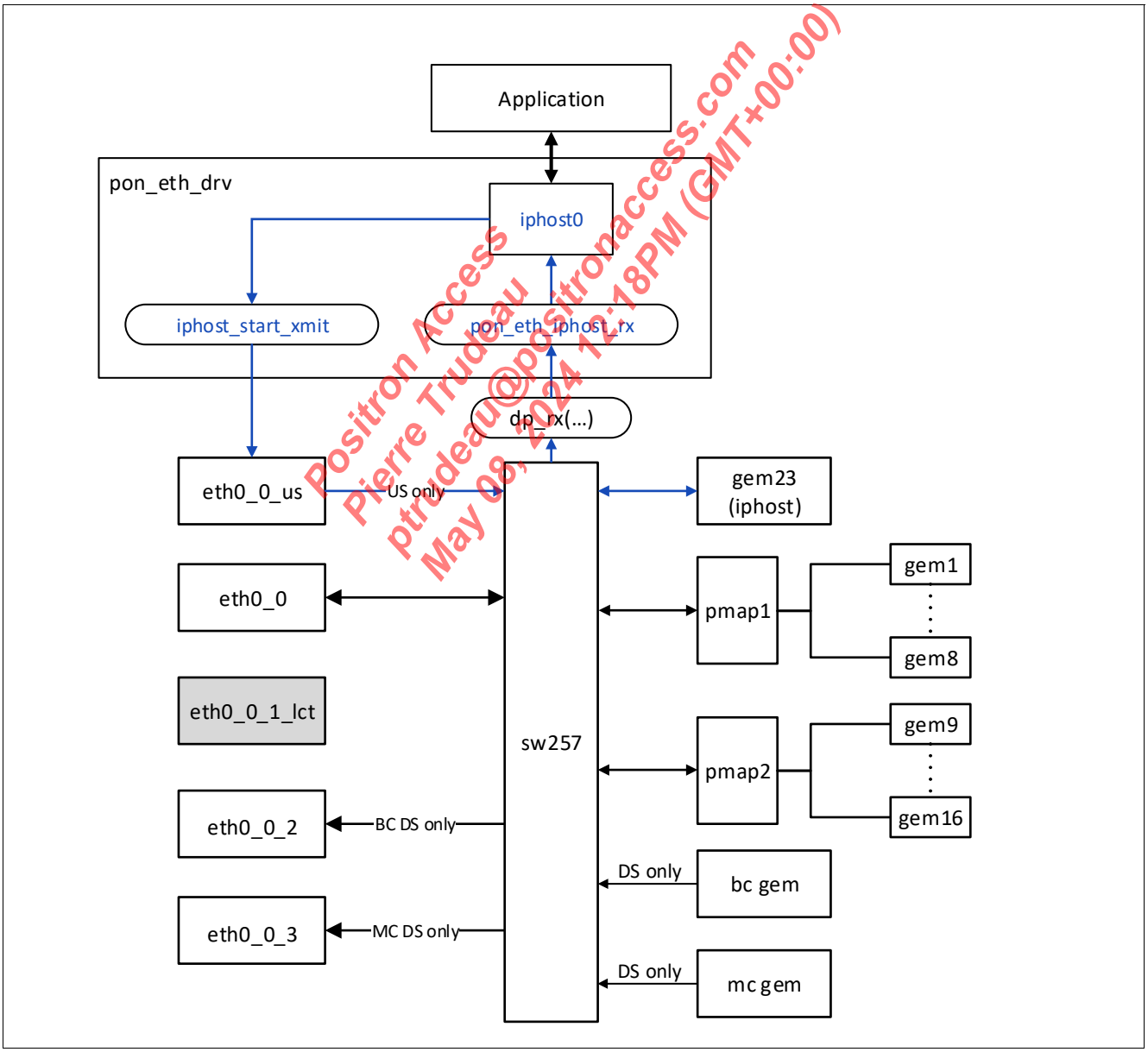


Figure 33 IP Host Network Interface View

Figure 34 shows the hardware flow of VoIP (IP host) data packets together with the general CPU bound packet forwarding flow.

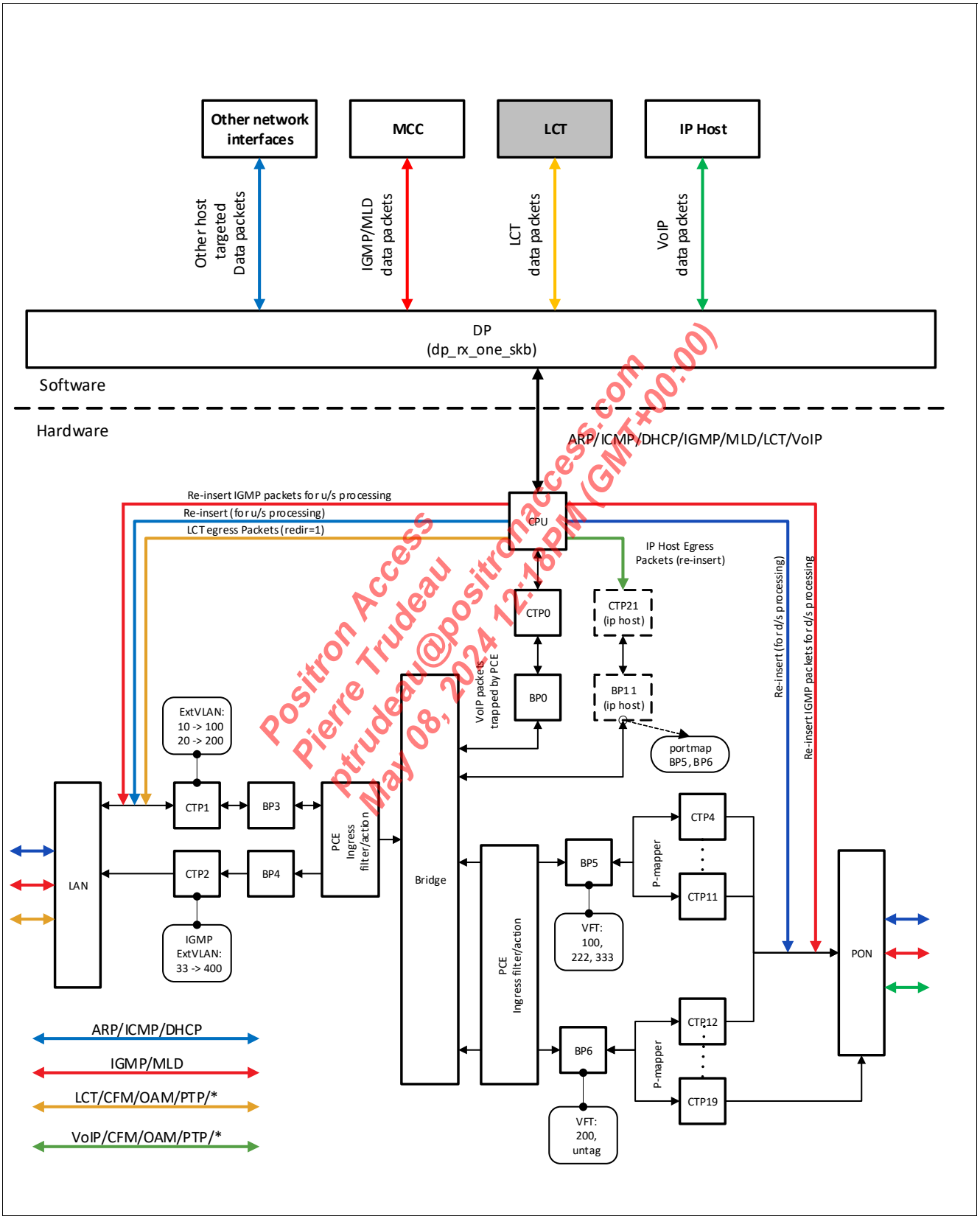


Figure 34 CPU Data Flow

3.17 IP Host Connected to a Bridge Port

Figure 35 shows the network device view of an example OMCI datapath configuration with IP Host. This implementation of the IP Host is valid starting from 10G PON Chipset System Package 1.16 and enables more than one instance of the IP Host interface to be created.

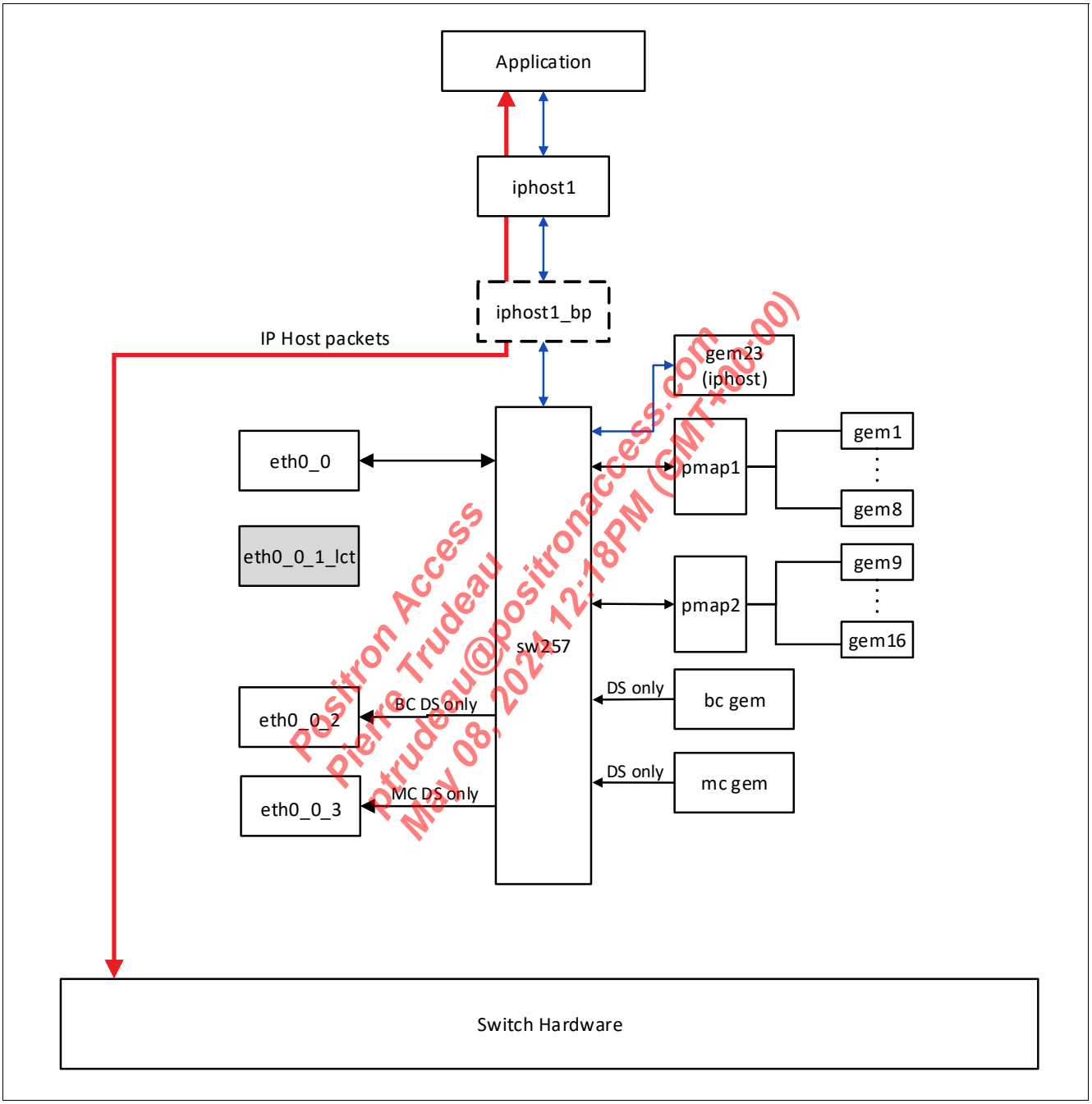


Figure 35 Networking Devices Used for the IP Host

Two network devices are used to support the IP Host connection to a bridge port. In the example above the are named *iphost1_bp* and *iphost1.* The *iphost1_bp* interface is connected to the Linux bridge where the rest of network devices used for forwarding the traffic are connected. This interface is mapped to a dedicated bridge port in the hardware and used for the configuration of the hardware bridge. The configuration is done using the Linux software bridge switchdev framework. This allows to configure the hardware bridge to forward IP Host traffic to the bridge port in hardware that is mapped to the *iphost1_bp* network device. The IP Host bridge port is a special port

and allows traffic to be received in the networking stack. It is independent of the CPU path which is followed by traffic trapped to Linux. The standard CPU path used different bridge port in hardware which is typically the first port with id zero.

The *iphost1_bp* is not used to handle any skbs in software, on this device no packets can be received or transmitted. It is used only for the bridge port hardware offload configuration. The packets coming to the software are passed to *iphost1 device* by the IP Host driver and then to networking stack. The user applications have to use this device to handle the IP Host traffic.

The packet received and transmit software flows are shown on **Figure 36**.

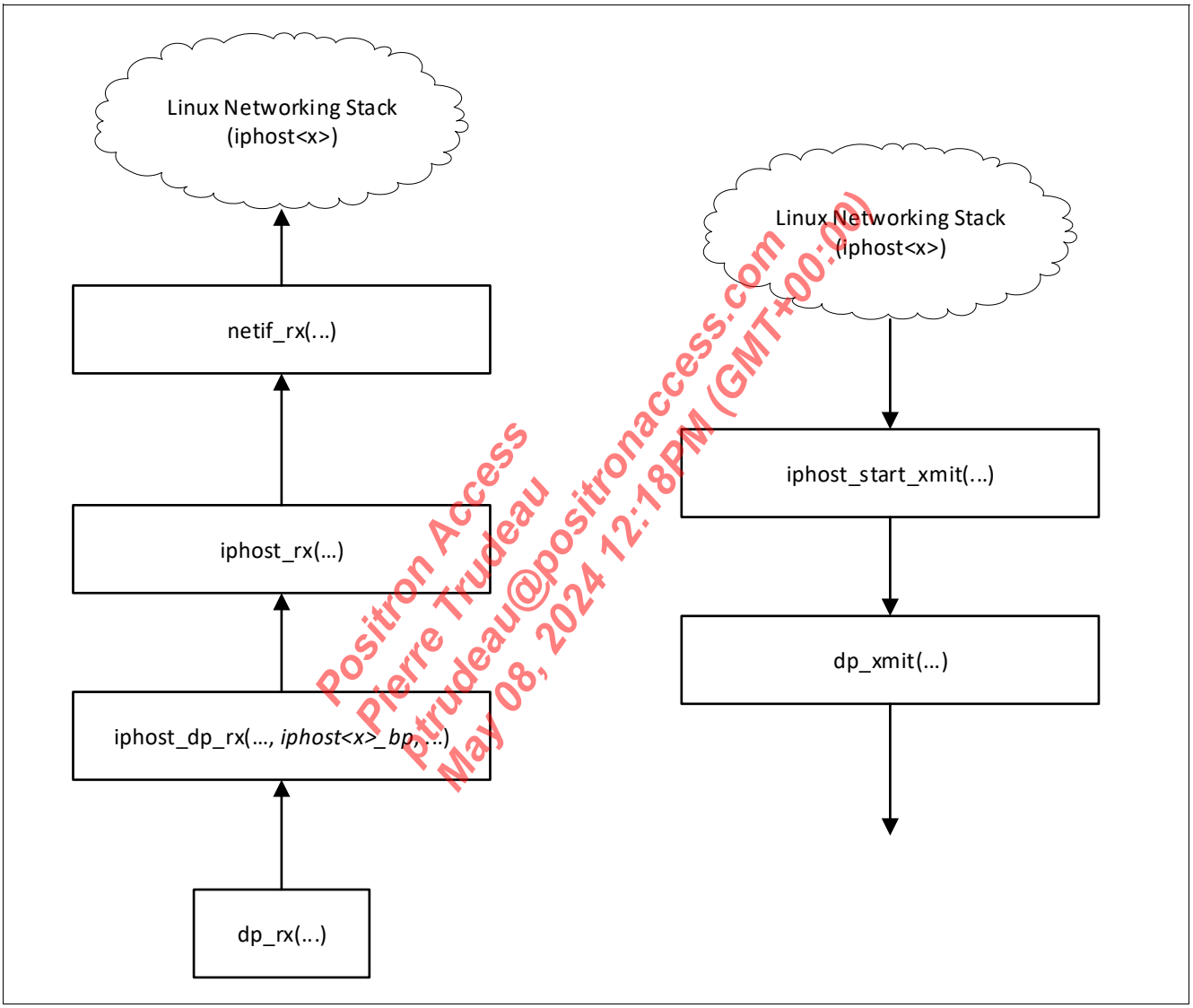


Figure 36 IP Host skbs Receive and Transmit Handling in Software

The receive path follows these steps:

1. The hardware switch forwards the IP Host traffic to the bridge port in hardware. The skb buff is received in *dp_rx(...)* function triggered by hardware interrupt. The skb contains all meta data describing all hardware ports which the packet was received on. The *dp_rx(...)* function is part of the Datapath Manager Driver.
2. The datapath driver call *iphost_dp_rx(...)*. The function has as argument the network device corresponding to the hardware bridge port on which the packet was received on. This callback is registered by the PON Ethernet driver. The function calls iphost_rx(...). No skb is assigned to *iphost<x>_bp* device and the skb is freed to avoid packets with ingress device *iphost<x>_bp* to reach the network stack.

3. The *iphost_rx(...)* forwards a cloned skb to a specific upper IP Host network device *iphost<x>*. The decision is based on the destination MAC address in the skb. Multicast and broadcast packets are forwarded to all upper *iphost<x>* devices. The *iphost_rx(...)* function sets the skb->dev pointer to the iphost<x> network device and calls netif_rx(...) to forward the packet to the Linux network stack.

The transmit path follows these steps:

1. An application sends a packet from *iphost<x>* network device.
2. Network stack call the i*phost_stack_xmit(...)* part of the *iphost<x>* network device ops.
3. *dp_xmit(...)* function is called with correctly filled skb meta data and datapath driver queues the packet for hardware transmission.
4. Packet is sent by the hardware according to the hardware switch bridging decision.

Create IP host devices by using these commands

```
root@ugwcpe:/# ip link add link ip0 iphost0_bp type iphost
root@ugwcpe:/# ip link add link iphost0_bp iphost0 type iphost
root@ugwcpe:/# ip -br l
lo               UNKNOWN        00:00:00:00:00:00 <LOOPBACK,UP,LOWER_UP>
ip6tnl0@NONE     DOWN           :: <NOARP>
sit0@NONE        DOWN           0.0.0.0 <NOARP>
dummy0           DOWN           c6:cd:58:84:0f:be <BROADCAST,NOARP>
ifb0             DOWN           36:a5:8b:72:f4:88 <BROADCAST,NOARP>
ifb1             DOWN           a2:34:c3:9e:dd:aa <BROADCAST,NOARP>
gre0@NONE        DOWN           0.0.0.0 <NOARP>
gretap0@NONE     DOWN           00:00:00:00:00:00 <BROADCAST,MULTICAST>
ip6gre0@NONE     DOWN           :: <NOARP>
teql0            DOWN           <NOARP>
VANI0            DOWN           26:ae:13:e7:99:2a <BROADCAST,MULTICAST>
VUNI0_0          UNKNOWN        d6:71:e9:36:52:40 <BROADCAST,MULTICAST,UP,LOWER_UP>
VUNI0_1          DOWN           92:2e:51:e0:c3:c1 <BROADCAST,MULTICAST>
VUNI0_us         DOWN           86:7f:c0:7b:92:01 <BROADCAST,MULTICAST>
VUNI0_2          DOWN           52:d3:8e:35:67:34 <BROADCAST,MULTICAST>
ins0             UNKNOWN        00:00:00:00:00:00
<BROADCAST,MULTICAST,NOARP,UP,LOWER_UP>
eth0_0           UP             02:9a:96:fb:c4:98 <BROADCAST,MULTICAST,UP,LOWER_UP>
eth0_0_1_lct     DOWN           02:54:77:09:4a:2a <BROADCAST,MULTICAST>
eth0_0_2         DOWN           c6:9f:d7:55:ab:ca <BROADCAST,MULTICAST>
eth0_0_3         DOWN           f6:2e:f3:28:a3:8d <BROADCAST,MULTICAST>
eth0_0_us        DOWN           1a:69:c2:08:a5:5a <BROADCAST,MULTICAST>
eth0_1           DOWN           d6:73:be:0d:2d:b2 <BROADCAST,MULTICAST>
eth0_1_1_lct     DOWN           36:f2:6e:a2:87:e2 <BROADCAST,MULTICAST>
eth0_1_2         DOWN           fe:03:ee:0d:db:5c <BROADCAST,MULTICAST>
eth0_1_3         DOWN           5e:65:d1:85:0f:0b <BROADCAST,MULTICAST>
eth0_1_us        DOWN           9e:26:51:7b:ba:46 <BROADCAST,MULTICAST>
pon0             DOWN           00:00:00:00:00:00 <BROADCAST,MULTICAST>
ip0              DOWN           00:00:00:00:00:00 <BROADCAST,MULTICAST>
br-lan           UP             02:9a:96:fb:c4:98 <BROADCAST,MULTICAST,UP,LOWER_UP>
tcont-omci@pon0  DOWN           62:40:c9:2c:ed:d0 <BROADCAST,MULTICAST,M-DOWN>
gem-omci@pon0    LOWERLAYERDOWN 02:50:f1:00:00:01 <NO-
CARRIER,BROADCAST,MULTICAST,PROMISC,UP,M-DOWN>
iphost0_bp@ip0   DOWN           da:fb:0b:d9:6f:b8 <BROADCAST,MULTICAST,M-DOWN>
iphost0@iphost0_bp DOWN           a2:c6:35:a0:6a:0e <BROADCAST,MULTICAST,M-DOWN>
```

3.18 LCT Network Interface

The OMCI LCT network interface is implemented in a similar way as the IP Host network interface. The LCT ingress packets from the LAN port are trapped by programming special PCE filter rules in the switch hardware. They are configured by using tc flower filters and tc trap action on the `eth0_0_x_lct` network device. The rules currently used are:

```
root@prx321-sfu-lab-pon:/# tc filter show dev eth0_0_1_lct ingress
filter protocol ipv6 pref 2160 flower
filter protocol ipv6 pref 2160 flower handle 0xc0000870 classid :fffe
 eth_type ipv6
  ip_proto icmpv6
  icmp_type 135
  skip_sw
        action order 1: gact action trap
         random type none pass val 0
         index 292 ref 1 bind 1

filter protocol 802.1Q pref 2168 flower
filter protocol 802.1Q pref 2168 flower handle 0xc0000871 classid :fffe
 vlan_ethtype ipv6
  eth_type ipv6
  ip_proto icmpv6
  icmp_type 135
  skip_sw
        action order 1: gact action trap
         random type none pass val 0
         index 293 ref 1 bind 1

filter protocol ipv6 pref 2176 flower
filter protocol ipv6 pref 2176 flower handle 0xc0000880 classid :fffe
 eth_type ipv6
  ip_proto icmpv6
  icmp_type 136
  skip_sw
        action order 1: gact action trap
         random type none pass val 0
         index 294 ref 1 bind 1

filter protocol 802.1Q pref 2184 flower
filter protocol 802.1Q pref 2184 flower handle 0xc0000881 classid :fffe
  vlan_ethtype ipv6
  eth_type ipv6
  ip_proto icmpv6
  icmp_type 136
  skip_sw
        action order 1: gact action trap
         random type none pass val 0
         index 295 ref 1 bind 1

filter protocol all pref 4112 flower
filter protocol all pref 4112 flower handle 0xa0080801 classid :fffb
  dst_mac ac:9a:96:fb:c6:50
```

```
  skip_sw
        action order 1: gact action trap
         random type none pass val 0
         index 298 ref 1 bind 1

filter protocol arp pref 4116 flower
filter protocol arp pref 4116 flower handle 0xa0080800 classid :fffd
 eth_type arp
  skip_sw
        action order 1: gact action trap
         random type none pass val 0
         index 290 ref 1 bind 1

filter protocol 802.1Q pref 4117 flower
filter protocol 802.1Q pref 4117 flower handle 0xa0080801 classid :fffd
 vlan_ethtype arp
  eth_type arp
  skip_sw
        action order 1: gact action trap
         random type none pass val 0
         index 291 ref 1 bind 1

filter protocol [35063] pref 4124 flower
filter protocol [35063] pref 4124 flower handle 0x1 classid :fff9
 eth_type 88f7
  skip_sw
        action order 1: gact action trap
         random type none pass val 0
         index 415 ref 1 bind 1

filter protocol 802.1Q pref 4125 flower
filter protocol 802.1Q pref 4125 flower handle 0x1 classid :fff9
 vlan_ethtype [35063]
  eth_type 88f7
  skip_sw
        action order 1: gact action trap
         random type none pass val 0
         index 416 ref 1 bind 1
```


In the rules, the first nibble of the classid defines the traffic class assigned to the different packet types. The traffic class is used to select the CPU ingress queue. The packet types currently trapped to the LCT are these:

Packets carrying the `eth0_0_x_lct` MAC address as destination MAC, ARP packets, NDP neighbor solicitation and advertisement and precision time protocol (PTP) packets.

Figure 37 shows the exact LCT data path handling. The rules trapping the LCT packets to the CPU match after the packets go through the extended VLAN processing. This requires special handling when extended VLAN drop rule is configured. In case of untagged drop rules, the packets are not dropped but reassigned to a new bridge port in hardware with empty port map. The packets matching the previous rules are trapped to the CPU and the packets not matching are dropped.

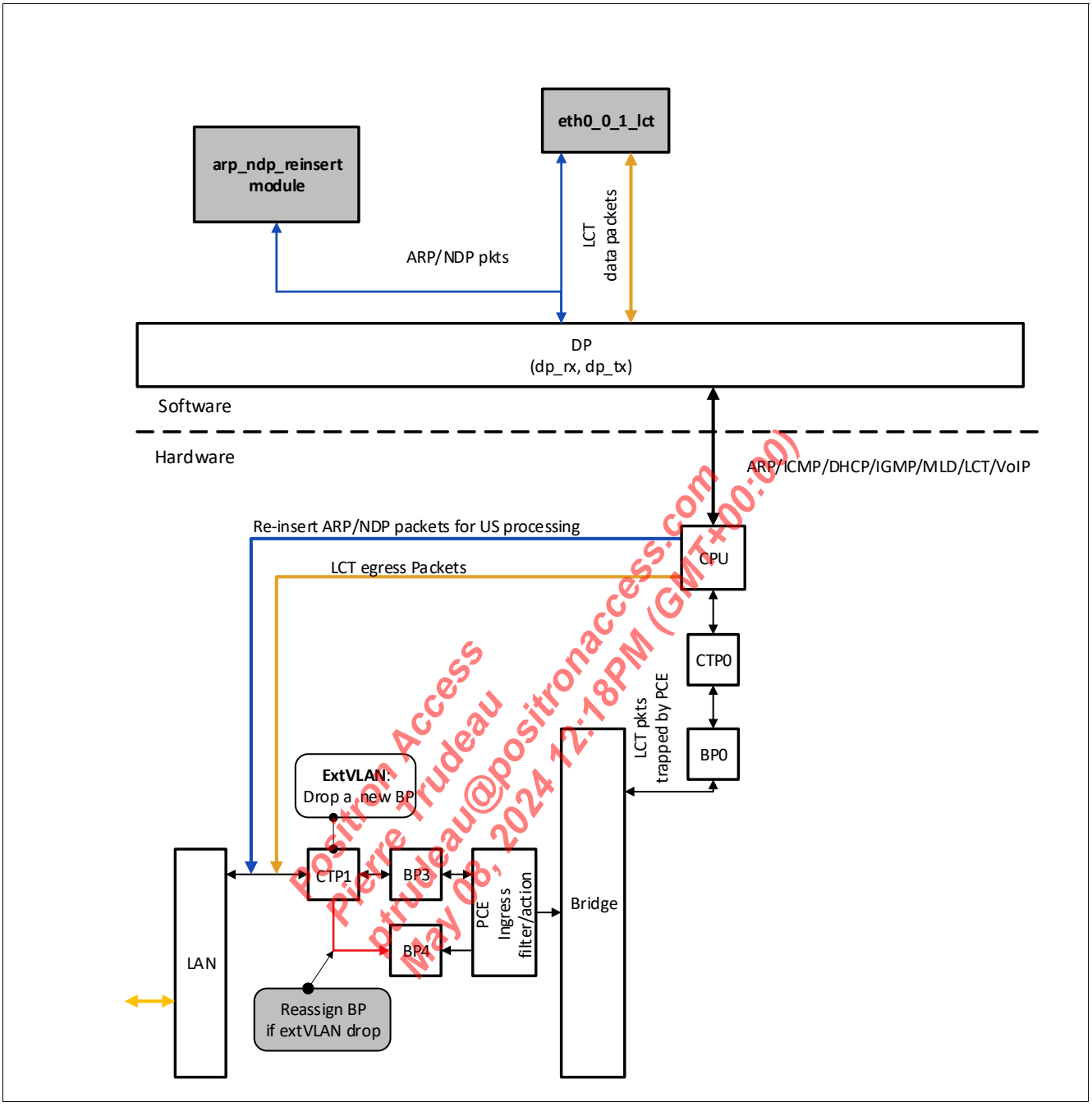


Figure 37 LCT Data Path

When the ARP/NDP packets reach the software, one copy of the packets is forwarded to the LCT network device and another is received in the netfilter hook of the `arp_ndp_reinsert` module. The module reinserts these packets back for hardware processing and the packets are forwarded upstream, according to the data path configuration.

Default LCT Trap Rules

The default configuration of the trap rules for the LCT interfaces is defined in `pon_net_netlink.c` of the `pon_netlib` library. Check for the function `netlink_lct_trap_set()`. Vendor-specific changes must be applied here, when required.

3.19 Special Tag Functionality

The special tag is a Layer 2 extension to the Ethernet header, identified by a dedicated Ethertype value. It is used to override the forwarding and QoS functionality of the switch on the ingress side and to provide additional frame-status information on the egress side.

This is used when external switches or similar devices are connected to a UNI port which requires additional information to determine the target output port.

The special tag is also used between the PON IP hardware module and the switching hardware to communicate PON-specific information, such as GEM port ID and allocation ID.

The special tag is identified by the special Ethertype located after the source MAC address in the frame. This allows the transmission of the frame via an Ethernet network to a remote receiver. The special tag content has a fixed length of 6 byte. For internal communication or point-to-point communication, it can be configured per egress port when the frame contains additional content in place of the Ethertype. This option is only available in egress direction, as in ingress direction an Ethertype is always expected. **Figure 38** illustrates this functionality.

The pause frames generated by the MAC do not contain a special tag. This can result in a mix of frames with and without special tag on one egress port. The frames with a special tag which do not use the special Ethertype are distinguished from pause frames since the first bit after the MAC addresses is 0 for frames with a special tag while the pause frames have an Ethertype of 0x8808.

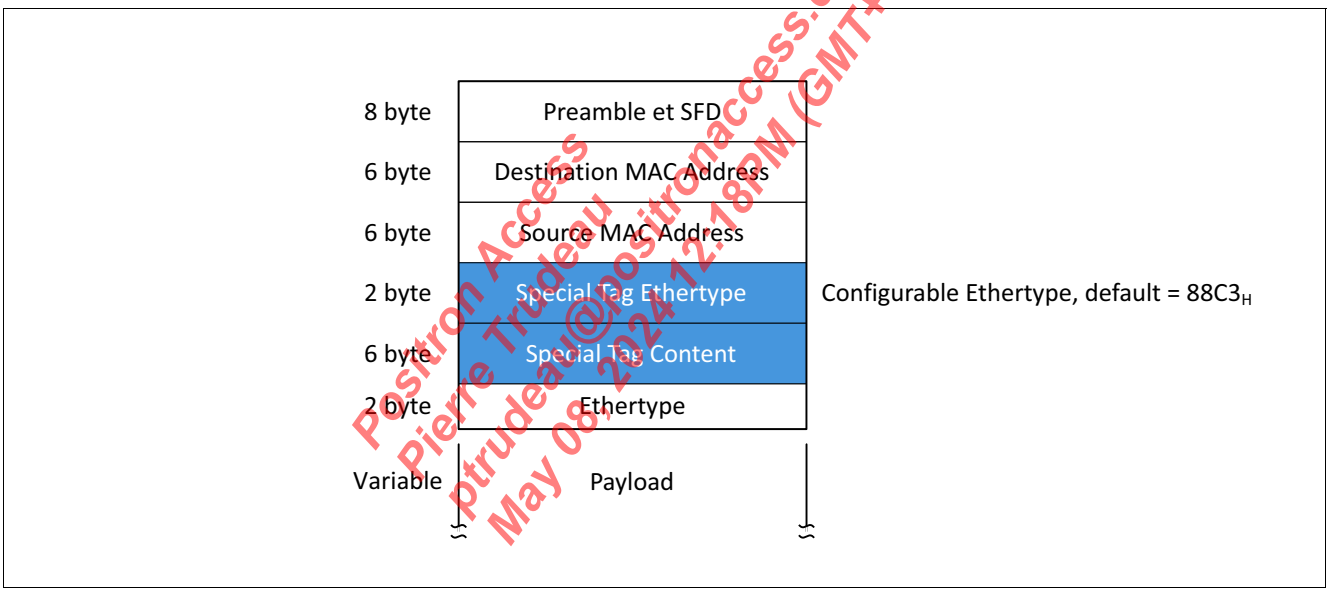


Figure 38 Special Tag Location in the Frame

Figure 39 Special Tag Format

The special tag content format is different for the ingress and the egress. The different tag formats are described in **Chapter 3.19.1** and **Chapter 3.19.2**.

3.19.1 Ingress Special Tag

The ingress special tag is used to override the classification function and the default frame forwarding of the integrated switch (GSWIP). The special tag detection on ingress is either enabled or disabled per port. When the detection is disabled, the frame containing a special tag is treated as a regular frame and the content of the frame is ignored. By default, the special tag detection is disabled.

When the ingress special tag detection is enabled, the content of the tag is used for the frame forwarding decision. The ingress special tag must always have a special tag Ethertype. **Table 20** details the special tag content.

Note: The bytes are defined as MSB = bit 7 and LSB = bit 0. Byte 0 is the first byte received and transmitted.

Table 20 Special Tag Ingress Format

Byte	Bit	Description
0	[7:0]	Ethertype MSB (byte 1) (configurable, default: 88_H)
1	[7:0]	Ethertype LSB (byte 2) (configurable, default: $C3_H$)
2	[7:4]	Reserved
2	[3:0]	Traffic Class Target traffic class. This field is valid only when traffic class enable is set.

Table 20 Special Tag Ingress Format (cont'd)

Byte	Bit	Description
3	[7:6]	Reserved
3	[5]	Forced Learning Disable (LNMD) Forced leaning disable control 0_B **Ignore** Learning enable or disable is determined by the GSWIP configurations. 1_B **Disable** Learning is forced to be disabled regardless of the GSWIP configurations.
3	[4]	Traffic Class Enable (TC_EN) Traffic class enable 0_B **Ignore** Traffic class is determined by the GSWIP configurations. 1_B **Enable** Traffic class is determined in the special tag traffic class field.
3	[3:0]	Reserved
4	[7:6]	Reserved
4	[5]	Time Stamp Flag (PTP) This packet marks whether a packet is a PTP message triggering the packet time recording. 0_B **Disable** Not a time stamp event packet 1_B **Enable** Time stamp event packet
4	[4:0]	Source/Destination Sub-Interface ID Bit 12 to 8 MSB field of source/destination sub-interface ID of the packet. For non-PCE-bypass packet with INS=1, it carries the source or destination sub-interface ID, depending on the insertion point. For non-PCE-bypass and INS=0, it carries the source sub-interface ID. For PCE-bypass traffic and EXT=0, it carries the destination sub-interface ID. For PCE-bypass traffic and EXT=1, it carries the source sub-interface ID.
5	[7:0]	Source/Destination Sub-Interface ID Bit 7 to 0 LSB field of source/destination sub-interface ID of the packet. For non-PCE-bypass packet with INS=1, it carries the source or destination sub-interface ID, depending on the insertion point. For non-PCE-bypass and INS=0, it caries the source sub-interface ID. For PCE-bypass traffic and EXT=0, it carries the destination sub-interface ID. For PCE-bypass traffic and EXT=1, it carries the source sub-interface ID.
6	[7:0]	Reserved
7	[7:0]	Reserved

Table 21 describes the encoding of the destination port map enable and traffic class enable fields of the special tag content.

Table 21 Traffic Class Map Coding

Traffic Class Enable (TC_EN)	Resulting Traffic Class
0	The traffic class is based on the classification result in switch.
1	The traffic class is taken from the special tag (classified Traffic Class ignored).

These are additional action flags in the ingress special tag.

- Time-stamp action. This action flag is used to trigger the latching of the time-stamps for the received and transmitted packet.
- Force learning disable. This action is used to disable the learning of the source MAC address of the received frame in the MAC bridging table.

3.19.2 Egress Special Tag

The egress special tag contains the internal switch status and debug information. The special tag transmission on egress is either enabled or disabled per egress logical port. When the egress special tag is disabled, no special tag is inserted in the egress frame. When the egress special tag function is enabled, each egress frame transmitted on that port contains the special tag.

Table 22 details the egress special tag format.

Note: The bytes are defined as MSB = bit 7 and LSB = bit 0. Byte 0 is the first byte transmitted and received.

Table 22 Special Tag Egress External Format (with Ethertype)

Byte	Bit	Description
0	[7:0]	Ethertype byte 1 (configurable, default: 88_H)
1	[7:0]	Ethertype byte 2 (configurable, default: $C3_H$)
2	[7:4]	Reserved
2	[3:0]	Traffic Class Traffic class of the packet determined by the GSWIP QoS classification.
3	[7:2]	Reserved This field is fixed to 0.
3	[1:0]	Packet Type (PKT_TYPE) 00_B **EtherNIPv6** Ethernet packet without IPv6 payload 10_B **EtherIPv6** Ethernet packet with IPv6 payload $X1_B$ **Reserved** Reserved
4	[7:6]	Reserved
4	[5]	Time Stamp Flag (PTP) This packet marks whether a packet is a PTP message which triggers the packet time recording. 0_B **Disable** Not a time stamp event packet 1_B **Enable** Time stamp event packet
4	[4:0]	Source/Destination Sub-Interface ID Bit 12 to 8 MSB field of the source/destination sub-interface ID of the packet. For EXT=0, it carries the destination sub-interface ID. For EXT=1, it carries the source sub-interface ID.
5	[7:0]	Source/Destination Sub-Interface ID Bit 7 to 0 LSB field of source/destination sub-interface ID of the packet. For EXT=0, it carries the destination sub-interface ID. For EXT=1, it carries the source sub-interface ID.
6	[7:0]	Reserved
7	[7:0]	Reserved

3.19.3 Special Tag Software Configuration

The special tag handling is configured by using the switch API.

Note: The internal GSWIP module always creates a special tag on egress and can receive a special tag on ingress. The MAC handling is configurable to adapt this to the outside world.

This is tested from the command line by using the `switch_cli` functions. The general command format is:

```
switch_cli gswss macop <2/3/4: MacIdx> <0/1 RX/TX> <0/1/2 FCS/SPTAG/TIME>
<0/1/2/3 Mode0/Mode1/Mode2/Mode3>
```

The operation modes are defined as:

- Egress direction (TX)
  - Mode 0: The packet is sent without a special tag, an existing special tag is not removed (transparent mode).
  - Mode 1: The existing special tag created by the GSWIP is replaced.
  - Mode 2: The existing special tag created by the GSWIP is not modified
  - Mode 3: The existing special tag created by the GSWIP is removed, the outgoing packet does not contain a special tag.
- Ingress direction (RX)
  - Mode 0: The packet received from outside does not have a special tag (from external) and a special tag is not inserted to the GSWIP.
  - Mode 1: The packet received from outside does not have a special tag (from external) and a special tag is inserted to the GSWIP.
  - Mode 2: The packet received from outside includes a special tag (from external) and a special tag is not inserted to the GSWIP.

The typical configuration are:

- Regular operation without special tags between the PRX device and a remote Ethernet device
  - Egress mode = 3
  - Ingress mode = 1
- Connection with an external switch, using special tags in both directions
  - Egress mode = 2
  - Ingress mode = 2

Note: The connection of GSW1xx devices requires a parser code modification of the integrated GSWIP due to a different special tag format.

3.20 MAC Security

The MAC Security (MACsec, [44]) function is supported towards the LAN side. A single 10G capable MACsec hardware module is supported by the device. Select one of the two LAN interfaces to enable MACsec.

The regular `ip` commands are used to handle the MACsec function.

3.20.1 MACsec Application Example

The commands in this section demonstrate the MACsec functionality. The higher layer software that is needed to handle authentication and encryption key handling is not included in the SDK.

This example uses:

```
mm:mm:mm:mm:mm:mm = Station 1 MAC address
nn:nn:nn:nn:nn:nn = Station 2 MAC address
11111111111111111111111111111111 = key 1 (32 hex characters = 128 bit)
22222222222222222222222222222222 = key 1 (32 hex characters = 128 bit)
```

Adapt the values of MAC addresses, IP addresses, and encryption keys if needed.

Station 1 – ONU

The PRX-based ONU (“Station 1”) is configured as:

```
ip link add link eth0_0 macsec0 type macsec encrypt on
ifconfig macsec0 hw ether mm:mm:mm:mm:mm:mm
ip macsec add macsec0 tx sa 0 pn 1 on key 01 11111111111111111111111111111111
ip macsec add macsec0 rx port 1 address nn:nn:nn:nn:nn:nn
ip macsec add macsec0 rx port 1 address nn:nn:nn:nn:nn:nn sa 0 pn 1 on key 01
22222222222222222222222222222222
ip link set macsec0 up
ip addr add 192.168.3.1/24 dev macsec0
```

Station 2 – Linux PC

The PC as peer station (“Station 2”) is configured as:

```
ip link add link eth0 macsec0 type macsec encrypt on
ifconfig macsec0 hw ether nn:nn:nn:nn:nn:nn
ip macsec add macsec0 tx sa 0 pn 1 on key 01 22222222222222222222222222222222
ip macsec add macsec0 rx port 1 address mm:mm:mm:mm:mm:mm
ip macsec add macsec0 rx port 1 address mm:mm:mm:mm:mm:mm sa 0 pn 1 on key 01
11111111111111111111111111111111
ip link set macsec0 up
ip addr add 192.168.3.2/24 dev macsec0
```

After the configuration has been applied to both stations, it is possible to exchange traffic. To check the counters, use this command:

```
ip -s macsec show
```

3.21 Single GEM Port Mapping to Multiple Queues

The mapping of a single GEM to multiple queues is required when the traffic management option in the **ONU-G** is configured as *Rate controlled upstream traffic*. The data path must be configured to enable the mapping of the GEM traffic to multiple queues based on the VLAN priority (pbit). The QoS engine lookup table mode must be set to mode 3 which maps the traffic based on GEM index (subif) and traffic class. The traffic class assignment happens on the UNI CTP ingress port. Each VLAN priority (pbit) is mapped to the corresponding traffic class. **Figure 40** shows how the complete data path is configured.

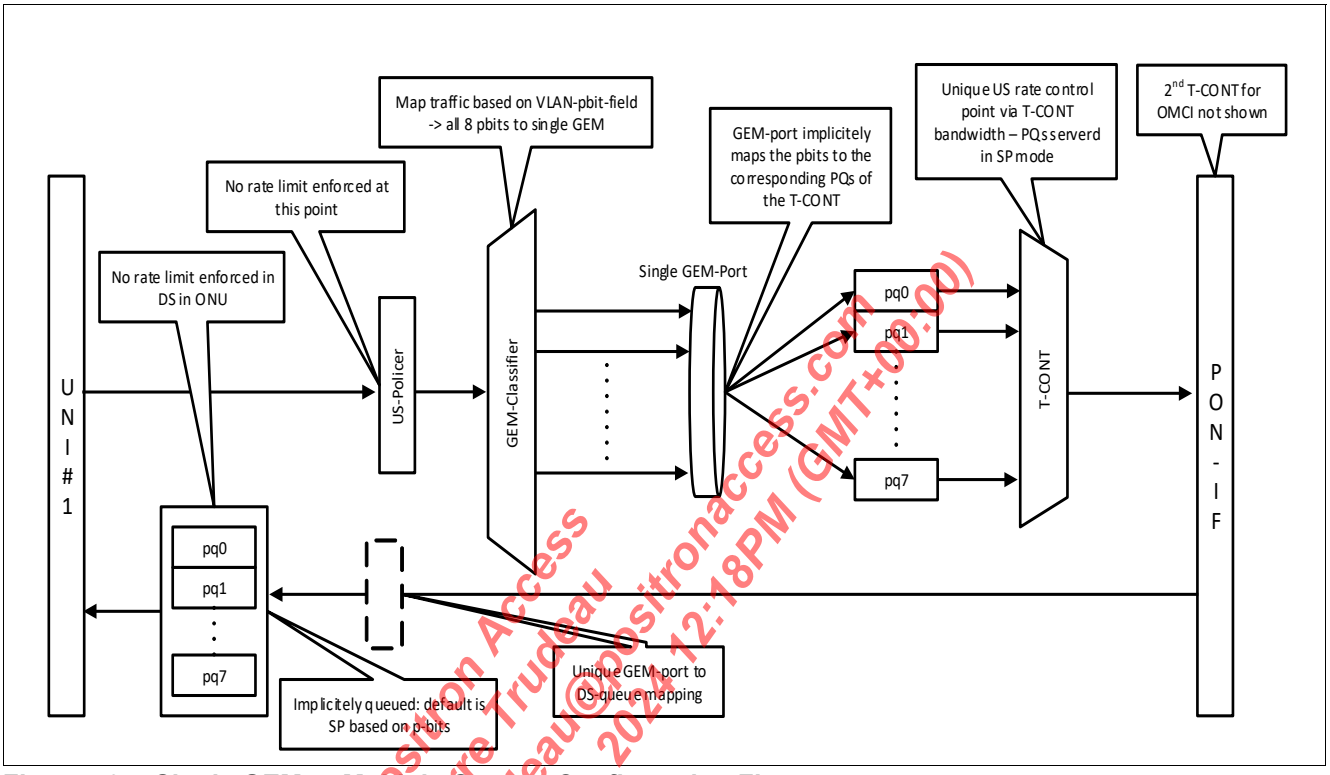


Figure 40 Single GEM to Multiple Queues Configuration Flow

These are example commands for the lookup mode and QoS configuration. This configuration must be done by the OMCI stack depending on the traffic management option.

```
$ ip link set pon0 type pon queue_lookup_mode 3

$ tc qdisc add dev tcont32768 root handle 5: prio bands 8 priomap 7 6 5 4 3 2 1 0
$ tc qdisc add dev tcont32768 handle ffff: clsact
$ tc filter add dev tcont32768 ingress pref 1 flower skip_sw indev gem1023 classid
5:1 action ok cookie 0007
$ tc filter add dev tcont32768 ingress pref 2 flower skip_sw indev gem1023 classid
5:2 action ok cookie 0006
$ tc filter add dev tcont32768 ingress pref 3 flower skip_sw indev gem1023 classid
5:3 action ok cookie 0005
$ tc filter add dev tcont32768 ingress pref 4 flower skip_sw indev gem1023 classid
5:4 action ok cookie 0004
$ tc filter add dev tcont32768 ingress pref 5 flower skip_sw indev gem1023 classid
5:5 action ok cookie 0003
$ tc filter add dev tcont32768 ingress pref 6 flower skip_sw indev gem1023 classid
5:6 action ok cookie 0002
```

```
$ tc filter add dev tcont32768 ingress pref 7 flower skip_sw indev gem1023 classid
5:7 action ok cookie 0001
$ tc filter add dev tcont32768 ingress pref 8 flower skip_sw indev gem1023 classid
5:8 action ok cookie 0000
```

Use the tc action cookie option to pass by the traffic class. The GEM network devices define the GEM port index (subif).

3.22 Data Path OAM APIs

The `dp_oam` driver provides kernel space APIs used to program the hardware to extract the OAM packets to the CPU based on different packet attributes such as the OAM packet header level and code fields, VLAN ID, port, and others. The driver is located at `drivers/net/datapath/dp_oam` in the kernel sources. The APIs are defined in `dp_oam.h`. The packets can be extract at four points by hardware: Before and after queuing, with or without the VLAN modification. The API configures the flow table in the switch hardware. Each call to the `dp_oam_act_add(...)` creates a rule in this table. The rules in this table have a pattern that matches some attributes in the packet like the ones previously described and an action is executed if the pattern matches. The rules in this case use the special OAM extraction action. Because the flow table is used for other purposes as well, such as VLAN forwarding and forwarding traffic to the CPU, the OAM stack must take care of the correct ordering of the OAM extraction rules by providing the correct priority in respect to the other non-OAM and OAM rules.

The driver provides a debugfs interface[1)] used to try the API functionality. Use this command to get usage information from the interface:

```
$ echo help > /sys/kernel/debug/dp_oam/dbg
[43381.235964] usage: echo add/del {args} > /sys/kernel/debug/dp_oam/dbg
[43381.241003] args:
[43381.242851] global - rule will configured in global table
[43381.248275] drop - drop rule will be configured instead of extraction
[43381.254688] dev {ifindex} - rule will on specified subif, on port if global is set
[43381.262239] ext_pt {val} - Extraction point
[43381.266405] level {val} - OAM level
[43381.269877] opcode {val} - OAM opcode
[43381.273523] vlan_id {val} - VLAN id
[43381.276995] prio {val} - Rule priority
[43381.280729] index {val} - Index to be removed
```

The addition and deletion rules are tested here. It is possible to list the rules using `cat` on the previous file.

If extraction points are used to forward traffic to the CPU, the extracted packets have to be mapped to one of the available CPU queues depending on the required priority. This has to be done for both the upstream and the downstream direction. Example tc commands are listed below:

```
# Upstream queue mapping
$ tc filter add dev tcont32768 egress pref 2323 protocol all flower skip_sw indev
pon0 classid 1006:1 action ok
# Downstream queue mapping
$ tc filter add dev eth0_0 egress pref 1313 protocol all flower skip_sw indev pon0
classid 1004:1 action ok
```

The *classid* defines the priority queue to which the extracted packets are mapped.

The network device that receives the OAM packet can differ based on the extraction point. There are always two to trap the OAM packets to the software:

- Using normal bridge forwarding functionality
- Using extraction points

Figure **Figure 41** shows possible receiving network devices. The example is for CFM packets ingressing in downstream direction.

The CFM packets network device destination in case of extraction points two and three after queuing is the UNI device. In the example this is *eth0_0*. The same is valid for upstream, in this case the device is either a pmapper or a gem port network device.

1) The debug file system (debugfs) is intended for trial use only and shall not be included in productive software.

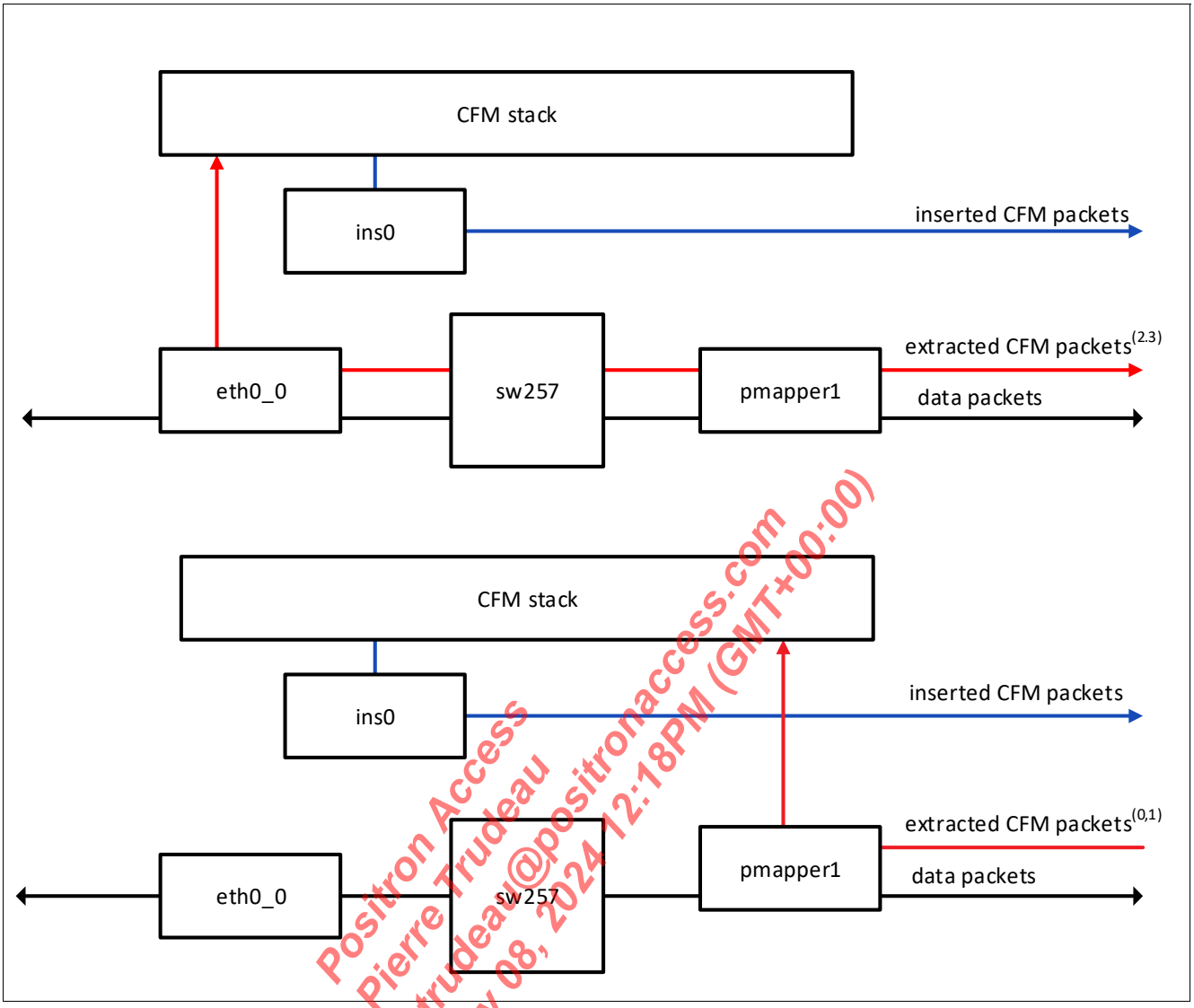


Figure 41 CFM Packets Destination Network Devices Depending on Extraction Type

The CFM packets network device destination in case of bridge forwarding or extraction points (0,1) before queuing is either a pmapper or a gem network device in the example this is *pmapper1*. In upstream this is one of the UNI devices *eth0_0*.

There are currently two extraction points for the cases before and after queuing. The difference is that the first one is without the VLAN modification and the second is with the VLAN modification. The kernel APIs support all cases. The user-space APIs support only the second one.

The transmission of CFM packets can be done in two different ways:

- Packet re-insertion: In this case the packets pass the complete hardware processing.
- Packet normal transmission: In this case the packets is enqueued directly into the egress queue for transmission. The software is responsible to do all the necessary packet modifications.

The applications need to use the special *"ins0"* network device to re-insert packets for hardware processing. This works in the same way as the multicast control packets handling as described in **Chapter 3.10**.

3.22.1 General UpMEP Handling Example

An upMEP can be implemented both in user space or kernel space, depending on the OAM stack implementation. **Figure 42** shows how a CFM stack can use the hardware support to implement the packet filtering in hardware. The stack has to configure rules to filter OAM packets based on their OAM PDU MEG level. In general two rules are needed to achieve the desired filtering for an upMEP. One rule is needed to trap the MEG level that is to be handled and one rule to drop the packets with lower levels that shall pass through transparently. The higher levels are forwarded by default by the hardware.

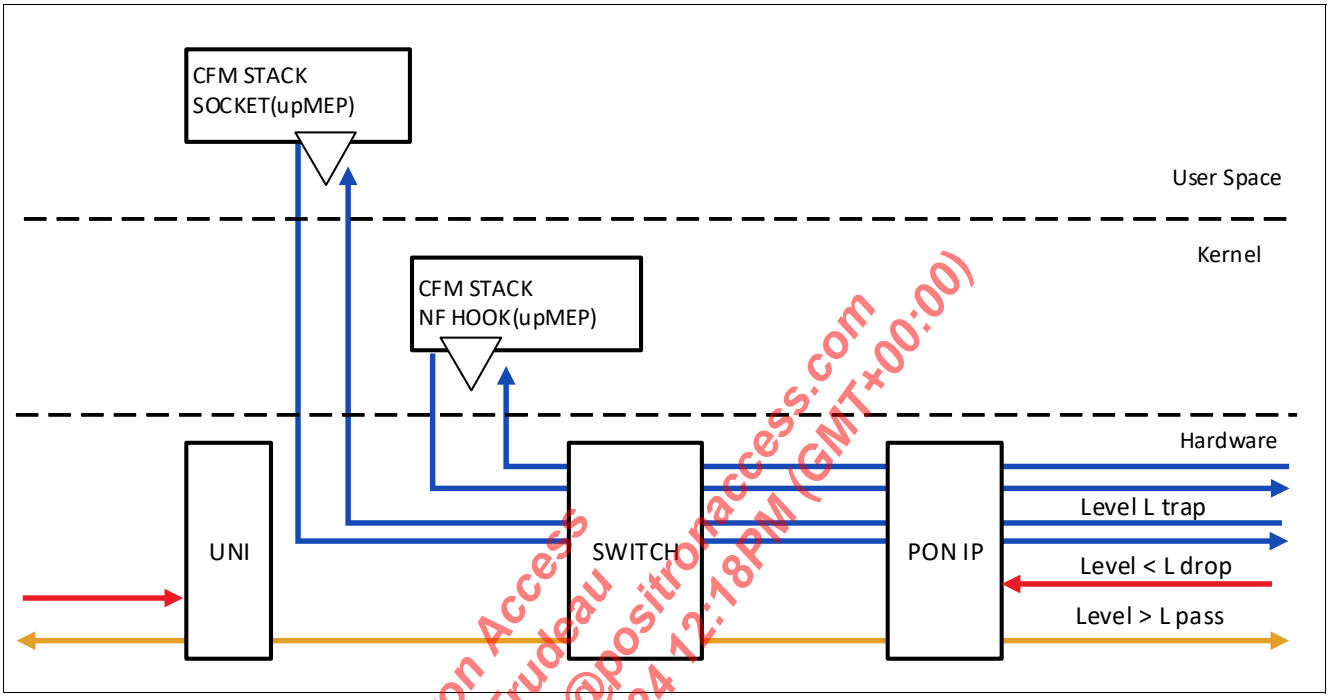


Figure 42 UpMEP Packet Handling Example

The OAM PDU filter for an upMEP can be created by using these commands:

```
# map extracted packets to 4th CPU queue
tc filter add dev eth0_0 egress pref 1313 protocol all flower skip_sw indev pon0
classid 1004:1 action ok
# trap CFM level=5 to CPU for netdev with index 22 (pmapper1)
echo add dev 22 vlan_id 587 level 5 ext_pt 3 > /sys/kernel/debug/dp_oam/dbg
# drop CFM level < 5 for netdev with index 22 (pmapper)
echo add dev 22 vlan_id 587 level 5 drop > /sys/kernel/debug/dp_oam/dbg
# drop CFM level <= 5 for netdev index 2 (eth0_0)
echo add dev 2 vlan_id 587 level 6 drop > /sys/kernel/debug/dp_oam/dbg
```

The CFM stack kernel implementation has to use the APIs from the *dp_oam* module to manage these filtering rules.

```
dp_oam_act_add(struct dp_oam_args *args)
dp_oam_act_del(struct dp_oam_args *args)
```

The *struct dp_oam_args* has to be filled with the required parameters similar to the examples above. For example, the network device index has to be assigned in the *ifindex* field and the VLAN ID in the *vlan_id* field. Global rules can also be configured. The global rules are not assigned to any specific ingress interface and match on all interfaces.

The CFM OAM stack can be implemented either in the kernel or in the user-space as shown in **Figure 42**.

The CFM OAM stack in kernel space has to implement a netfilter hook where the received packets are processed and the necessary replies are generated and sent back. The transmission is done by using the `dp_xmit(…)` function. It supports both re-insertion and sending packets directly to the egress queue. The OAM stack is also responsible for the correct ordering of the filtering rules. The *dp_oam_args* has a priority argument. The stack is also responsible to take care if different processing contexts are needed like timers or kernel threads.

The implementation of the CFM OAM stack in the user-space has to follow the same rules. The main difference is that is has to use the Linux socket APIs for reception and transmission. The CFM OAM packets reaching the CPU are received usually on network devices in a Linux bridge. This can lead to packet duplications if *RAW* sockets are used. In this case special filtering in the software needs to be done. One possibility is to extend the *mcast_bridge_filter* driver to filter the CFM packets instead of forwarding them.

The *debugfs* is only for testing purposes and shall not be used to configure the level filtering. The *dp_oam* module APIs must be used for kernel space implementations.

3.22.2 MEP Counter Insertion in LMM and LMR OAM PDUs

Counter values must be inserted into LMM (Loss Measurement Message) and LMR (Loss Measurement Reply) messages.

The examples show MEP configurations using the *debugfs* which must be used only for the API verification. They can be used as reference to see how the internal kernel *dp_oam* APIs can be used, especially which parameters must be supplied. It is highly recommend to check the code of the driver for more details.

3.22.2.1 Downstream UpMEP (PON to LAN)

An example debugfs command configuration for counter insertion into ingressing LMM and egressing LMR PDUs is shown below. These rules have to be configured before the more generic rules that are matching only on a distinct level.

The counter insertion requires two hardware rules in the flow engine per direction. One rule is for the counters itself and one rule is for matching the LMM/LMR PDUs and inserting the current counter value into these packets.


```
# map extract packets to 4th CPU queue
tc filter add dev eth0_0 egress pref 1313 protocol all flower skip_sw indev pon0
classid 1004:1 action ok

# create ingress counter for VLAN 587 (upstream)
echo add dev 2 vlan_id 587 counter prio 2 > /sys/kernel/debug/dp_oam/dbg
# drop CFM level <= 5 for netdev index 2 (eth0_0)
echo add dev 2 vlan_id 587 level 6 drop prio 1 > /sys/kernel/debug/dp_oam/dbg
# setup counter insertion for tx packets
echo add dev 2 vlan_id 587 opcode 42 level 5 rmon 1 rmon_tx rmon_offset 14 prio 3 >
/sys/kernel/debug/dp_oam/dbg

# create ingress counter for VLAN 587 (downstream)
echo add dev 22 vlan_id 587 counter prio 2 > /sys/kernel/debug/dp_oam/dbg
# drop level < 5
echo add drop dev 22 vlan_id 587 level 5 prio 1 > /sys/kernel/debug/dp_oam/dbg
# setup packet etraction with counter insertion for rx
echo add dev 22 vlan_id 587 opcode 43 level 5 ext_pt 3 rmon 2 rmon_tx rmon_bypass
rmon_offset 26 prio 3 > /sys/kernel/debug/dp_oam/dbg
# trap CFM level < 5 for netdev with index 22 (pmapper)
echo add dev 22 vlan_id 587 level 5 ext_pt 3 prio 4 > /sys/kernel/debug/dp_oam/dbg
```

3.22.2.2 Upstream UpMEP (LAN to PON)

```
# map extract packets in us to 6th CPU queue
tc filter add dev tcont32768 egress pref 2323 protocol all flower skip_sw indev pon0
classid 1006:1 action ok
echo add dev 2 vlan_id 587 counter > /sys/kernel/debug/dp_oam/dbg
echo add dev 22 vlan_id 587 counter > /sys/kernel/debug/dp_oam/dbg
# setup packet etraction with counter insertion
echo add dev 2 vlan_id 587 opcode 43 level 5 ext_pt 3 rmon 1 rmon_tx rmon_bypass
rmon_offset 26 > /sys/kernel/debug/dp_oam/dbg
echo add dev 2 vlan_id 587 level 5 drop prio 1 > /sys/kernel/debug/dp_oam/dbg
# trap CFM level < 5 for netdev with index 22 (pmapper)
echo add dev 2 vlan_id 587 level 5 ext_pt 3 prio 4 > /sys/kernel/debug/dp_oam/dbg
# setup counter insertion for tx packets
echo add dev 22 vlan_id 587 opcode 42 level 5 rmon 2 rmon_tx rmon_offset 14 >
/sys/kernel/debug/dp_oam/dbg
echo add dev 22 vlan_id 587 level 6 drop prio 1 > /sys/kernel/debug/dp_oam/dbg
```

3.22.2.3 Downstream DownMEP (PON to LAN)

```
# map extract packets to 4th CPU queue
tc filter add dev eth0_0 egress pref 1313 protocol all flower skip_sw indev pon0
classid 1004:1 action ok

# create ingress counter for VLAN 587 (upstream)
echo add dev 2 vlan_id 587 counter prio 2 > /sys/kernel/debug/dp_oam/dbg
# drop CFM level <= 5 for netdev index 2 (eth0_0)
echo add dev 2 vlan_id 587 level 6 drop prio 1 > /sys/kernel/debug/dp_oam/dbg
# setup counter insertion for tx packets
echo add dev 2 vlan_id 587 opcode 42 level 5 rmon 1 rmon_tx rmon_bypass rmon_offset
30 prio 3 > /sys/kernel/debug/dp_oam/dbg

# create ingress counter for VLAN 587 (downstream)
echo add dev 22 vlan_id 587 counter prio 2 > /sys/kernel/debug/dp_oam/dbg
# drop level < 5
echo add drop dev 22 vlan_id 587 level 5 prio 1 > /sys/kernel/debug/dp_oam/dbg
# setup packet etraction with counter insertion for rx (ext_pt 2, rx counter)
echo add dev 22 vlan_id 587 opcode 43 level 5 ext_pt 1 rmon 2 rmon_offset 10 prio 3
> /sys/kernel/debug/dp_oam/dbg
# trap CFM level=5 for netdev with index 22 (pmapper)
echo add dev 22 vlan_id 587 level 5 ext_pt 1 prio 4 > /sys/kernel/debug/dp_oam/dbg
```


3.22.2.4 API Counter Types, Offsets, and PDU Types

The *dp_oam* C API parameters can again be easily derived from the examples above. The packet offset values for the counter insertion are very important. The recommend offsets are shown in **Table 23**. The counter offset depends on the counter type. For example, the *"TX bypass counter"* offset is relative to the start of the packet and the *"TX counter"* offset is relative to the end of MAC header, before the Ethertype.

Table 23 LMM and LMR Counter Type and Offset Usage

Type Description	Counter	Offset
UpMEP ingress LMR	TX bypass	1 (end of packet)
UpMEP egress LMR	TX	(offset_of_TxFCb + ethtype_size) = 12 + 2 = 14
UpMEP ingress LMM	TX bypass	(eth_hlen + vlan_hlen*tags + offset_of_RxFCf = 14 + 4*tags + 8
UpMEP egress LMM	TX	(offset_of_TxFCb + ethtype_size) = 4 + 2 = 6
DownMEP ingress LMR	RX	1 (end of packet)
DownMEP egress LMR	TX bypass	eth_hlen + vlan_hlen*tags + offset_of_RxFCf = 14 + 4*tags + 8
DownMEP ingress LMM	RX	(offset_of_RxFCf + ethtype_size) = 8 + 2 = 10
DownMEP egress LMM	TX bypass	eth_hlen + vlan_hlen*tags + offset_of_RxFCf = 14 + 4*tags + 8

Note: The offsets are from the start of the OAM PDU header starting after the 0x8902 Ethertype. The value of the RxFCl counter is appended to the end of ingressing LMR OAM PDUs.

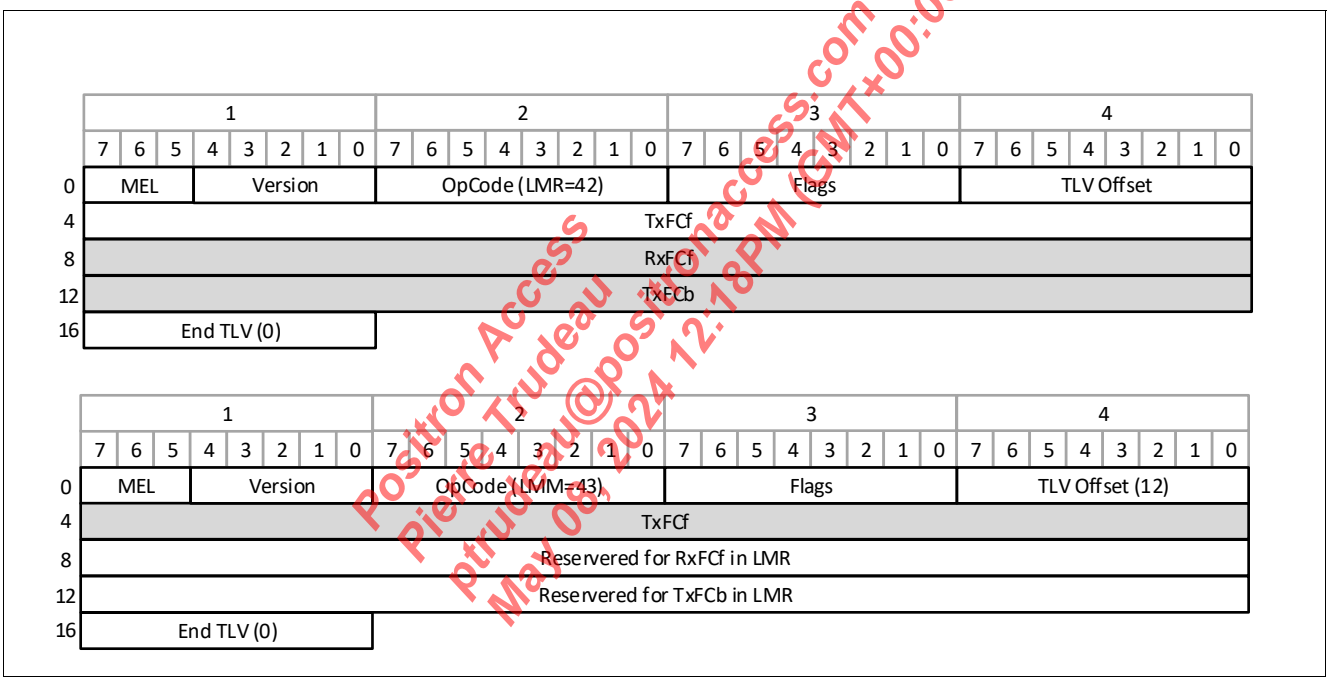


Figure 43 CFM OAM LMR and LMM PDU Format

CCM Counter Insertion

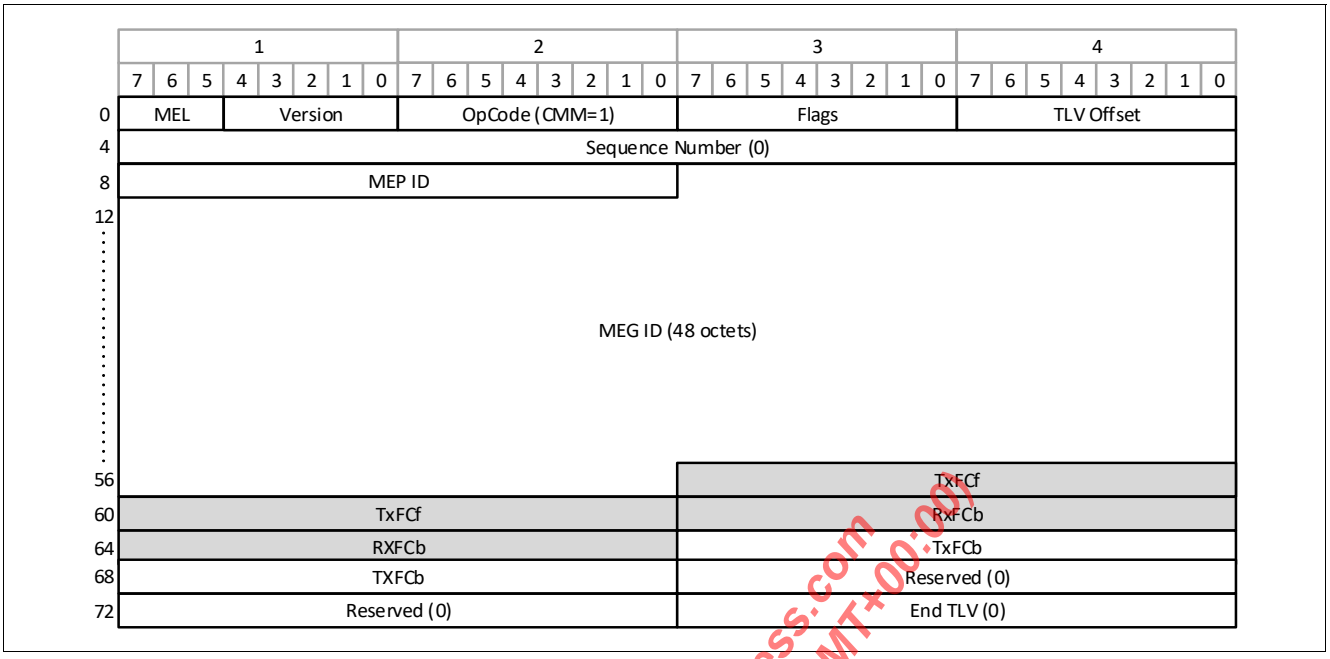


Figure 44 CCM OAM PDU Format

The OAM API configuration parameters for the counter insertion into CCM OAM PDUs can be derived from the *debugfs* configuration commands.

```
# CCM counter insertion
tc filter add dev eth0_0 egress pref 1313 protocol all flower skip_sw indev pon0
classid 1004:1 action ok
# tx counter
echo add dev 2 vlan_id 699 counter prio 2 > /sys/kernel/debug/dp_oam/dbg
# setup counter insertion for tx packets
echo add dev 2 vlan_id 699 opcode 1 level 5 rmon 1 rmon_tx rmon_offset 60 prio 3 >
/sys/kernel/debug/dp_oam/dbg
# rx counter
echo add dev 22 vlan_id 699 counter prio 3 > /sys/kernel/debug/dp_oam/dbg
# setup packet etraction with counter insertion for rx
echo add dev 22 vlan_id 699 opcode 1 level 5 ext_pt 3 rmon 2 rmon_tx rmon_bypass
rmon_offset $((14+4+62)) prio 3 > /sys/kernel/debug/dp_oam/dbg
```


Table 24 LMM and LMR Counter Type and Offset Usage

Type Description	Counter	Offset
UpMEP ingress CCM	TX bypass	(eth_hlen + vlan_hlen*tags + offset_of_RxFCb) = 14 + 4*tags + 62 = 80
UpMEP egress CCM	TX	(offset_of_TxFCf + ethtype_size) = 58 + 2 = 60
DownMEP ingress CCM	RX	(offset_of_RxFCb + ethtype_size) = 62 + 2 = 64
DownMEP egress CCM	TX bypass	(eth_hlen + vlan_hlen*tags + offset_of_TxFCf = 14 + 4*tags + 58 = 76

3.22.3 MEP Timestamp Insertion into DMM, DMR, and 1DM OAM PDUs

3.22.3.1 UpMEP Downstream Timestamp Insertion into DMM and DMR OAM PDUs

The debugfs provides commands for the timestamp insertion into ingress DMM and egressing DMR OAM PDUs. The API parameters can be easily derived from the example.

```
# Map extracted packets to 4th CPU queue
tc filter add dev eth0_0 egress pref 1313 protocol all flower skip_sw indev pon0
classid 1004:1 action ok
# Insert timestamp into DMM RxTimeStampf field (DS)
echo add dev 22 vlan_id 699 opcode 47 level 5 ext_pt 3 ts_offset 14 timestamp
ts_format 2 > /sys/kernel/debug/dp_oam/dbg
# Insert timestamp into DMR TxTimeStampb field (US)
echo add dev 2 vlan_id 699 opcode 46 level 5 ts_offset 22 timestamp ts_format 2 >
/sys/kernel/debug/dp_oam/dbg
# Insert timestamp into DMR RxTimeStampb field (DS)
echo add dev 22 vlan_id 699 opcode 46 level 5 ext_pt 3 timestamp ts_offset 30
ts_format 2 > /sys/kernel/debug/dp_oam/dbg
# Insert timestamp into DMM TxTimeStampf field (US)
echo add dev 22 vlan_id 699 opcode 46 level 5 timestamp ts_offset 6 ts_format 2 >
sys/kernel/debug/dp_oam/dbg
```

Table 25 shows the recommended offsets for the different delay measurement OAM PDUs. See Figure 45 for more details and refer to the offsets of the different timestamp fields.

Table 25 DMM and DMR Timestamp Offset Usage

Type Description	Offset
ingress DMM	(offset_of_RxTimeStampf + ethtype_size) = 12 + 2 = 14
egress DMM	(offset_of_TxTimeStampf + ethtype_size) = 4 + 2 = 6
ingress DMR	(offset_of_RxTimeStampb + ethtype_size) = 28 + 2 = 30
egress DMR	(offset_of_TxTimeStampb + ethtype_size) = 20 + 2 = 22

Note: The offsets are from the start of the OAM PDU header starting after the 0x8902 Ethertype. This means that two bytes have to be added to each PDU field offset.

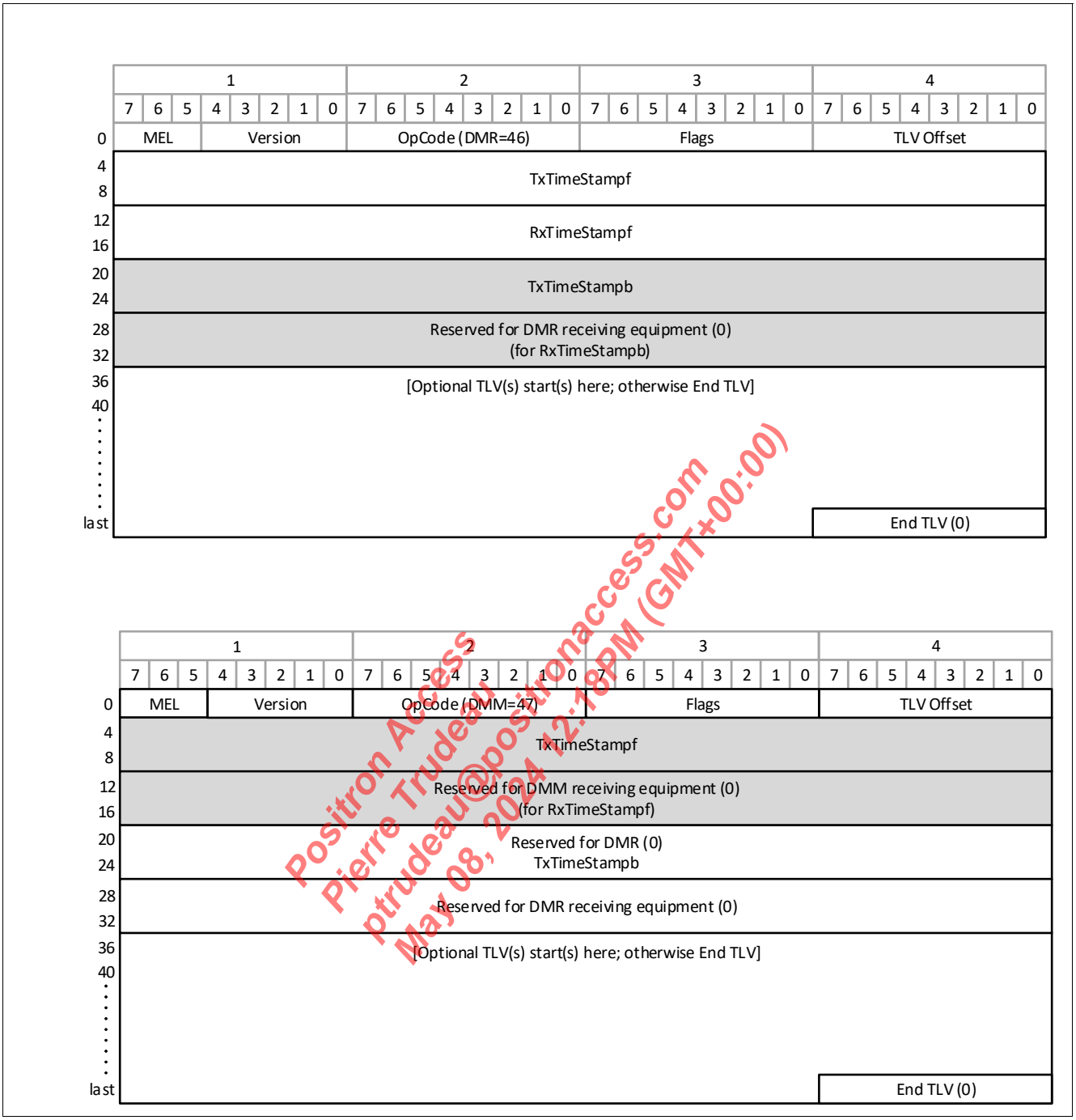


Figure 45 DMM and DMR OAM PDUs Format

3.22.3.2 Timestamp Insertion into 1DM OAM PDUs

The *dp_oam API parameters* for counter insertion into 1DM OAM PDUs packets can be derived from the commands below.

```
tc filter add dev eth0_0 egress pref 1313 protocol all flower skip_sw indev pon0
classid 1004:1 action ok
# Insert RxTimeStampf TS into 1DM (rx)
echo add dev 22 vlan_id 699 opcode 45 level 5 ext_pt 3 ts_offset 14 timestamp
ts_format 2 > /sys/kernel/debug/dp_oam/dbg
```

```
# Insert TxTimeStampf TS into 1DM (tx)
echo add dev 2 vlan_id 699 opcode 45 level 5 ts_offset 6 timestamp ts_format 2 >
/sys/kernel/debug/dp_oam/dbg
```

Table 26 1DM Timestamp Offset Usage

Type Description	Offset
ingress 1DM	(offset_of_RxTimeStampf + ethtype_size) = 12 + 2 = 14
egress 1DM	(offset_of_TxTimeStampf + ethtype_size) = 4 + 2 = 6

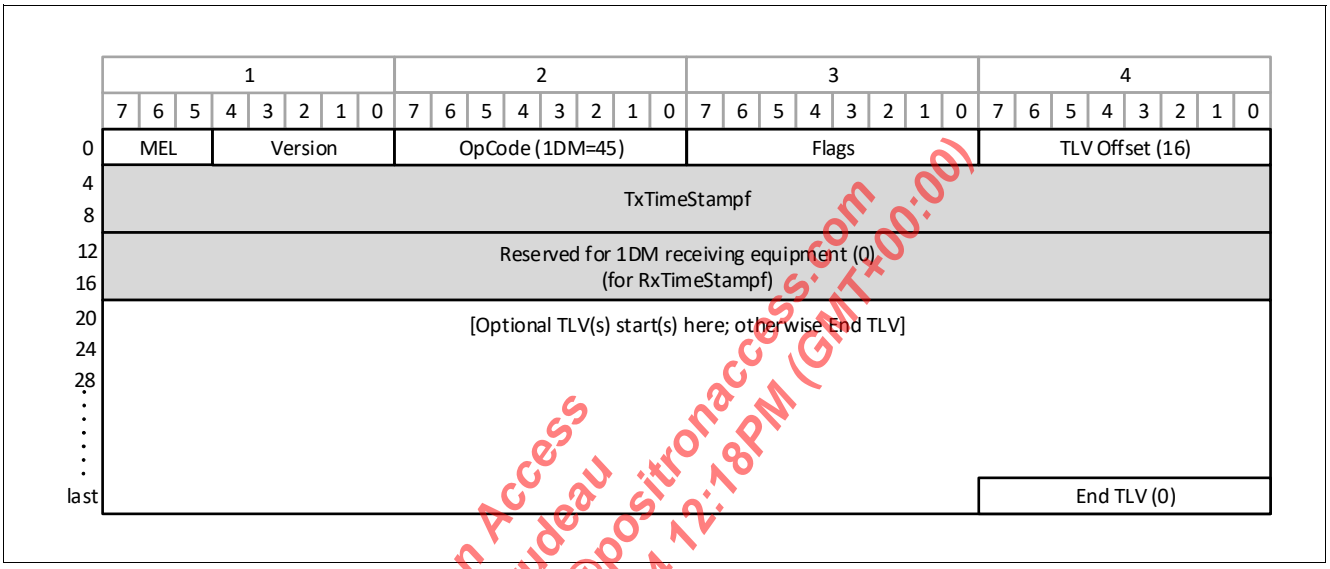


Figure 46 1DM OAM PDU Format

4 OMCI Management Stack

This chapter describes the implementation of an OMCI management stack for ITU-based PON systems.

The OMCI management protocol is defined by ITU-T G.988 [56] to control ONUs. It is applicable to various ITU-based PON technologies such as G-PON, XG-PON and XGS-PON, and NG-PON2.

The protocol is used to control an ONU remotely from an OLT located in a central office or street cabinet. The scope of OMCI management is to control all network functionality required to provide one or more user network interfaces (UNI), which are the demarcation point between the operator network and the user network.

The management protocol is modeled as a set of managed entities, where each hold a set of attributes. The number of attributes per managed entity is limited to 16 due to the transmission protocol structure. The size of each of the attributes is defined within the managed entity to which it belongs and can be a single byte, multiple bytes, or even a complex table.

Attention: The OMCI application delivered with the 10G PON Chipset System Package is intended to show how the lower layer software is connected to a vendor-specific OMCI software stack. This must be used for evaluation purposes only and is not ready to be included in productive systems.

4.1 OMCI Interoperability Aspects

Due to the flexibility of the OMCI standard and its evolution over time, there are a number of OLT installations in the field not able to use the latest OMCI enhancements and sometimes use vendor-specific enhancements which interfere with implementations that work well for other OLT types.

To be able to address specific interoperability requirements, the software provides a number of compile switches and UCI configurations to adapt the function to specific requirements (see Section 4.2 and Chapter 2.4.3.7.2, Table 6, OMCI Interoperability Settings).

By default, these options are disabled and must be set explicitly, when required. The handling of these options is controlled through the OMCI daemon start-up options. This allows adaptation of an already compiled image during system start-up.

4.1.1 OMCC Versions

The OMCI functionality has been increased over time, with each evolutionary step being indicated by an increasing version number. The numbering scheme differentiates between support for the baseline message format only (0xAn) and support for both baseline messages and extended messages (0xBn). The lower numbers (0x8n/0x9n) are related to earlier versions defined by ITU-T G.984.4, which are outdated and must not be used for new installations. These are not explicitly covered by the software implementation.

Table 27 provides a reference between the OMCC version numbers and the related OMCC standard versions.

Table 27 OMCC Version Numbering

OMCC Version Number	OMCI Document Version	Notes
Baseline Message Support		
0xA0	ITU-T G.988 (2010)	
0xA1	ITU-T G.988 (2010) + Amd. 1 (2011)	
0xA2	ITU-T G.988 (2010) + Amd. 2 (2012)	
0xA3	ITU-T G.988 (2012)	
0xA4	ITU-T G.988 (2012) + Amd. 1 (2014)	

Table 27 OMCC Version Numbering (cont'd)

OMCC Version Number	OMCI Document Version	Notes
Extended Message Support		
0xB0	ITU-T G.988 (2010).	Extended message support can be used by the OMCI implementation if downstream packet padding is provided by the OLT to a size of 48 bytes for messages which are shorter by definition. Upstream padding can optionally be enabled by selecting an interoperability configuration.
0xB1	ITU-T G.988 Amd.1 (2011)	
0xB2	ITU-T G.988 Amd.2 (2012)	
0xB3	ITU-T G.988 (2012)	
0xB4	ITU-T G.988 Amd1 (2014)	

Note: The definition and usage recommendation of further code points is under control of ITU.

4.2 OMCI Daemon

The OMCI software is implemented as a software daemon, which is started during the system start-up procedure. The start-up script `/etc/init.d/omcid.sh` is used to handle this. The UCI configuration values used by this script are located at `/etc/config/omci`.

4.2.1 OMCI Daemon Command Parameters

The parameters that can be applied when starting the OMCI daemon are listed below. The options are displayed through the `--help` and `--usage` input commands.

```
/# omcid --help
Usage: omcid [OPTION...]

  -c, --console               Start console
  -d, --debug_level=number    Default debug level for all modules (0 - max level
                              .. 4 - no output)
  -e, --emergency-stop        PLOAM emergency stop state

  -f, --running-path=path     Specify running config path (not supported)
  -g, --lct-port=port-number Specify UNI index to be used as local debug port
                              (1..4)
  -G, --lct-disable           lct disabled (down) by default during creation
  -i, --iop-mask=mask         IOP option mask
  -I, --omcc-if-mac=mac-address   Specify MAC address of interface for OMCI
                              communication
  -l, --log=path              Specify log file
  -M, --omcc-mac=mac-address Specify MAC address of ONU device to be managed
  -o, --omcc=version          Select OMCC version
  -O, --omcc-if=interface-name   Specify netdev interface for OMCI
                              communication
  -p, --config-path=path      Specify custom config path
  -w, --warmstart             Enable warmstart
  -?, --help                  Give this help list
      --usage                 Give a short usage message
  -V, --version               Print program version

# omcid --usage
Usage: omcid [-ceGw?V]
            [-d number] [-f path] [-g port-number] [-i mask]
            [-I mac-address] [-l path] [-M mac-address] [-o version]
            [-O interface-name] [-p path] [--console] [--debug_level=number]
            [--emergency-stop] [--running-path=path] [--lct-port=port-number]
            [--lct-disable] [--iop-mask=mask] [--omcc-if-mac=mac-address]
            [--log=path] [--omcc-mac=mac-address] [--omcc=version]
            [--omcc-if=interface-name] [--config-path=path] [--warmstart]
            [--help] [--usage] [--version]
```

4.2.2 OMCI Daemon Start

The start procedure depends on the selected operation environment.

4.2.2.1 UGW and prplOS

The OMCI daemon is started by the command:

```
# /etc/init.d/omcid.sh start
```

and stopped by the command:

```
# /etc/init.d/omcid.sh stop
```

The OMCI daemon is restarted by the command:

```
# /etc/init.d/omcid.sh restart
```

4.2.2.2 RDK-B

The OMCI daemon is restarted by the command:

```
# systemctl restart omcid
```

See [23] for more details.

4.2.3 Start-up Scripts

These sections show start-up scripts examples. The latest versions delivered in the 10G PON Chipset System Package may differ slightly.

For the UCI configurations related to OMCI, see Section 2.4.3.7.

4.2.3.1 OMCI Start-up Script

This is an example of the OMCI daemon start-up script (as of 10G PON Chipset System Package 1.22.0).

```
# cat /etc/init.d/omcid.sh

#!/bin/sh /etc/rc.common
# shellcheck shell=dash

# Copyright (C) 2020 - 2023 MaxLinear, Inc.
# Copyright (C) 2019 - 2020 Intel Corporation
# Copyright (C) 2011 lantiq.com
# Copyright (C) 2011 OpenWrt.org

START=85
STOP=10
USE_PROCD=1
OMCID_BIN=/usr/bin/omcid

source $IPKG_INSTROOT/lib/pon.sh

# This function definitions are needed in case we are running this script on
# non-falcon board
pon_ploam_emergency_stop_state_get() {
        echo "0"
}
```

```
pon_lct_num_get() {
        # in case this is not overwritten, we don't have a LCT
        return
}

if [ -f $IPKG_INSTROOT/lib/pon.sh ]; then
        . $IPKG_INSTROOT/lib/pon.sh
fi

wait_for_overlay() {
        while ! grep overlayfs:/overlay /proc/self/mounts > /dev/null; do
                sleep 1
        done
}

is_flash_boot() {
        grep overlayfs /proc/self/mounts > /dev/null
}

start_service() {
        local aon_mode
        local stdout
        local stderr

        if [ "$(pon_ploam_emergency_stop_state_get)" = "1" ]; then
                ploam_emerg_stop_state=-e
        fi

        # by default the output of omcid is redirected to /dev/console (option 2)
        # read values from config to allow changes to this
        stdout="$(uci -q get omci.default.stdout)"
        [ -z "$stdout" ] && stdout=2
        stderr="$(uci -q get omci.default.stderr)"
        [ -z "$stderr" ] && stderr=2

        aon_mode="$(uci -q get optic.common.aon_mode)"
        [ -z "$aon_mode" ] && aon_mode=0

        if [ "$aon_mode" -ne 1 ]; then
                procd_open_instance
                procd_set_param env
LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/opt/pon/lib/:/opt/intel/usr/lib/"
                procd_set_param command ${OMCID_BIN} ${ploam_emerg_stop_state}
                procd_set_param respawn
                procd_set_param stdout $stdout
                procd_set_param stderr $stderr
                procd_close_instance
        fi
}
```

```
service_stopped() {
        #echo "Wait for omcid to exit"
        while pgrep -x ${OMCID_BIN} > /dev/null; do sleep 1; done
        #echo "omcid is gone"
}

start() {
        (
                # Wait until overlay is mounted.
                # This can take some time on first boot.
                is_flash_boot && wait_for_overlay

                rc_procd start_service "$@"
        ) &
}

# If mib_file or omci_iop_mask are defined in U-Boot, copy them into uci config.
# The LCT port is also detected from network config during boot.
# Finally we copy the mib_file to cur_mib_file in case of warmstart and it does
# not exist yet (TODO: This should be done by the omci daemon itself).
boot() {
        local tmp
        local lct_num
        local mib_file
        local cur_mib_file
        local warmstart

        tmp=$(fw_printenv -n mib_file 2>&-)
        if [ -f "$tmp" ]; then
                uci set omci.default.mib_file="$tmp"
        elif [ -f "/etc/mibs/$tmp" ]; then
                uci set omci.default.mib_file="/etc/mibs/$tmp"
        fi

        tmp=$(fw_printenv -n omci_iop_mask 2>&-)
        if [ -n "$tmp" ]; then
                uci set omci.default.iop_mask=$tmp
        fi

        lct_num=$(pon_lct_num_get)
        [ -n "${lct_num}" ] && uci set omci.default.lct_port="$((1 + lct_num))"

        uci commit omci

        # do not start daemon in case ponmbox node is not enabled
        [ "$(pon_mbox_node_status)" != "okay" ] && exit 1

        start "$@"
}
```

```
service_triggers()
{
        procd_add_reload_trigger "omci" "gpon" "optic"
}
```

4.2.3.2 SFU Start-up Script

This is an example of the general start-up script `/lib/pon.sh` for SFU applications that use the PRX120, PRX126, or PRX321 device.

```
#!/bin/sh
# Copyright (C) 2011 OpenWrt.org
# Copyright (C) 2011 lantiq.com
# Copyright (C) 2018 - 2020 Intel Corporation
# Copyright (c) 2022 Maxlinear Inc.

pon_dt_name() {
        local machine
        local name

        # use cpuinfo
       machine=$(awk 'BEGIN{FS="[ \t]+:[ \t]"} /machine/ {print $2}' /proc/cpuinfo)

        [ -z "$machine" ] && {
                # or directly the device tree
                [ -e /proc/device-tree/model ] && machine=$(cat /proc/device-
tree/model)
        }

        # use first word in lower case
        name=$(echo $machine | awk '{print tolower($1);}')

        # to stay backward compatible, these names must be upper case:
        case "$name" in
        mdu)
                name="MDU"
                ;;
        sfp)
                name="SFP"
                ;;
        esac

        echo "$name"
}

pon_board_name() {
        local name

        # take board name from cmdline
        name=$(awk 'BEGIN{RS=" ";FS="="} /boardname/  {print $2}' /proc/cmdline)
        # or use devicetree as fallback
        [ -z "$name" ] && name=$(pon_dt_name)
```

```
        echo "$name"
}

# return ponmbox node status
pon_mbox_node_status() {
        local node
        local node_paths="/proc/device-tree/ssx1_1@18000000/"
        local status=""

        for path in $node_paths; do
                node=$(ls -d "$path"ponmbox*) 2> /dev/null
                [ -n "$node" ] && [ -e "$node/status" ] && {
                        status=$(cat "$node/status")
                        break
                }
        done

        echo "$status"
}

# return the (board specific) number of lan ports
pon_get_number_of_lan_ports() {
        local num="1"

        case $(pon_board_name) in
        prx300* | prx126* | prx120*)
                num="1"
                ;;
        prx321*)
                num="2"
                ;;
        esac

        echo $num
}

# return the (board specific) default interface used for "lct"
pon_default_lct_get() {
        case $(pon_board_name) in
        prx300* | prx321* | prx126* | prx120*)
                echo "eth0_0_1_lct"
                ;;
        *)
                echo "eth0_0"
                ;;
        esac
}

pon_lct_num_get() {
        local ifname=$(uci -q get network.lct.ifname)
```

```
        local uni_num=$(uci -q get network.lct.uni_num)
        if [ -n "$uni_num" ]; then
                echo "`expr $uni_num \\- 1`"
                return
        fi
        case ${ifname} in
        eth0_*)
                ifname=${ifname#eth0_}
                ifname=${ifname%_1_lct}
                echo "${ifname}"
                return
                ;;
        esac
}

pon_sgmii_mode() {
        local retval="6" # 6 = 10G
        # TODO: Extend to define other speed modes
        echo $retval
}

pon_base_mac_get() {
        local mac_addr=$(awk 'BEGIN{RS=" ";FS="="} $1 == "ethaddr" {print
$2}' /proc/cmdline)
        [ -z "$mac_addr" ] && mac_addr='ac:9a:96:00:00:00'
        echo $mac_addr
}

_hex2mac() {
        # adding ":" after each 2 chars and remove the last one again
        echo $1 | sed -e 's/../&:/g' -e 's/:$//'
}

_mac_add_offset() {
        local mac=$1
        local offs=$(printf "%d\n" $2) # make sure this is decimal

        mac=$(echo "$mac" | tr -d ':') # to remove colons
        mac=$(printf "%d\n" 0x$mac)    # to convert to decimal
        mac=$(expr $mac + $offs)       # add offset
        mac=$(printf "%012x\n" $mac)   # to convert to hex again

        echo $(_hex2mac $mac)
}

pon_mac_get() {
        local mac_offset
        local mac_limit=6

        case "$1" in
        eth0_0_1_lct | eth0_1_1_lct | lct)
```

```
                mac_offset=0
                ;;
        host | wan | eth1)
                mac_offset=1
                ;;
        eth0_0 | eth0_0_[23])
                # subifs 2/3 are for MC/BC and can use same mac as main ifc
                mac_offset=2
                ;;
        eth0_1 | eth0_1_[23])
                # subifs 2/3 are for MC/BC and can use same mac as main ifc
                mac_offset=3
                ;;
        iphost)
                mac_offset=$((4+$2))
                ;;
        *)
                mac_offset=-1
                ;;
        esac

        if [ $mac_offset -ge 0 ]; then
                echo $(_mac_add_offset $(pon_base_mac_get) $mac_offset)
        fi

        if [ "$mac_offset" -ge "$mac_limit" ]; then
                echo "pon.sh[warning]: used mac_offset($mac_offset) over the
limit($mac_limit)" >&2
        fi
}

pon_oui_get() {
      echo $(pon_base_mac_get)| awk 'BEGIN{FS=":"} {printf "%s:%s:%s\n", $1,$2,$3}'
}

pon_ploam_emergency_stop_state_get() {
        local retval="0"
        local state=$(fw_printenv -n ploam_emergency_stop_state 2>&-)

        case "$state" in
        0 | 1)
                retval="$state"
                ;;
        esac
        echo $retval
}

pon_is_10g_platform() {
        # true
        return 0
}
```

```
pon_transceiver_eeprom_path() {
        local path=""
        local bus=0

        # check for new path via pon_mbox
        case "$1" in
        dmi)
                path="/sys/class/pon_mbox/pon_mbox0/device/eeprom51"
                ;;
        serial_id)
                path="/sys/class/pon_mbox/pon_mbox0/device/eeprom50"
                ;;
        esac
        [ -e "$path" ] && {
                echo "$path"
                return
        }

        # backward compatible path
        case $(pon_board_name) in
        prx321-sfu-lab*)
                bus=4
                ;;
        esac

        case "$1" in
        dmi)
                path="/sys/bus/i2c/devices/${bus}-0051/eeprom"
                ;;
        serial_id)
                path="/sys/bus/i2c/devices/${bus}-0050/eeprom"
                ;;
        esac
        [ -e "$path" ] && {
                echo "$path"
                return
        }
}
```

4.2.4 MIB Initialization

Upon OMCI startup, the MIB is populated based on an initialization file. The file is defined by the OMCI UCI file at `/etc/config/omci`. See **Chapter 2.4.3.7**, **Table 5** for more detail.

Upon the first start, the file is copied to the file `/etc/mibs/current_mib.ini` which is used upon subsequent OMCI daemon starts. This allows to add the OLT-controlled configurations and perform an OMCI warm start.

4.2.4.1 MIB Initialization File Format

The MIB initialization file uses an ASCII format according to this structure, one line per managed entity definition:

```
<exception><space><class id><space><instance id><space><attribute 1><space>…
                                                          <space><attribute 16>
```

For example:

```
# ONU-G
256 0 ABCD 00000000000000 00000000 2 0 0 0 0

# ONU2-G
257 0 00000000000000000000 0xA0 0 1 1 128 8 1 128 0 0x007F 0 0 10

# ONU data
2 0 0
```

If not all possible attributes are explicitly given, the unspecified attributes are assumed to be 0.

There are several exceptions that can be marked for a managed entity:

- A “hash” (#) character at any position in a line marks the subsequent characters in the line as comments that are not evaluated by the OMCI software during file read.
- If the class ID value is preceded by a question mark (“?”) and a space character, then this managed entity is excluded from the MIB upload process.
- If the class ID value is preceded by an exclamation mark (“!”) and a space character, then this managed entity is optional and failing to initialize it will not prevent the OMCI daemon from starting.
  This allows to use the same file for different applications where only a subset of the managed entities is supported.
- If the class ID value is preceded by the letter “T”, then this is a table attribute. This is only used in automatically generated MIB files that are used for warm start. The syntax is:
  `T <class ID> <instance id> <attribute index> <data>`

Only one exception code is possible per managed entity class/instance combination.

Note: The exception character must be followed by a space to be correctly identified by the file parser.

4.2.4.2 OMCI Warm Start

The OMCI warm start option allows the OMCI management software to re-create the OMCI MIB from the latest configuration that has been received from the OLT instead from the default MIB file as defined in the UCI configuration (`option mib_file`, see **Chapter 2.4.3.7.2**).

Instead the configuration is read from a temporary file that is stored each time the configuration is enhanced or updated by the OLT (`option cur_mib_file`, see **Chapter 2.4.3.7.2**).

The warm start function is enabled by setting the configuration `option warmstart '1'`.

4.3 Managed Entity Overview

Table 28 provides an overview of the managed entities that are supported by the PON Adapter (available or planned) as described in Chapter 8. A reference implementation means that OEM-specific code must be added. The managed entities are grouped by functions, aligned with ITU-T G.988 [56].

Note: The implementation status is related to system SDK release SYS_1.22.0.

Table 28 Managed Entity Implementation Overview

Managed Entity Name	ITU-T G.988 Ch.	PON Modes	Creation	ME Class	Imple-mentation	PON Adapter
Equipment Management						
ONU-G	9.1.1	All	ONU	256	Yes	Yes
ONU2-G	9.1.2	All	ONU	257	Yes	Yes
ONU Data	9.1.3	All	ONU	2	Yes	N/A
Software Image	9.1.4	All	ONU	7	Yes	N/A
Cardholder	9.1.5	All	ONU	5	Yes	N/A
Circuit Pack	9.1.6	All	ONU	6	Yes	N/A
ONU Power Shedding	9.1.7	All	ONU	133	Yes	N/A
Port Mapping Package	9.1.8	All	ONU	161	Reference	N/A
Equipment Extension Package	9.1.9	All	ONU	160	Reference	N/A
Protection Data	9.1.10	All	ONU	279	No	N/A
Equipment Protection Profile	9.1.11	All	OLT	159	No	N/A
ONU Remote Debug	9.1.12	All	ONU	158	Reference	N/A
ONU Dynamic Power Management Control	9.1.14	All	ONU	336	Yes	Yes
ONU3-G	9.1.15	TWDM /TDM	ONU	441	Reference	N/A
ONU Manufacturing Data	9.1.16	All	ONU	456	Yes	N/A
ONU Time Configuration	9.1.17	All	ONU	457	Reference	N/A
ONU Operational Performance Monitoring History Data	9.1.18	All	OLT	459	No	N/A
ONU4-G	9.1.19	TWDM /TDM	ONU	460	No	N/A
ANI and Traffic Management						
ANI-G	9.2.1	All	ONU	263	Yes	Yes
ANI2-G	9.2.24	All	ONU	467	No	N/A
T-CONT	9.2.2	All	ONU	262	Yes	Yes
GEM Port Network CTP	9.2.3	All	OLT	268	Yes	Yes
GEM Interworking Termination Point	9.2.4	All	OLT	266	Yes	Yes
Multicast GEM Interworking Termination Point	9.2.5	All	OLT	281	Yes	Yes
GAL Ethernet Profile	9.2.7	All	OLT	272	Yes	Yes
Priority Queue	9.2.10	All	ONU	277	Yes	Yes
Traffic Scheduler	9.2.11	All	ONU	278	Yes	Yes

Table 28 Managed Entity Implementation Overview (cont'd)

Managed Entity Name	ITU-T G.988 Ch.	PON Modes	Creation	ME Class	Imple-mentation	PON Adapter
Traffic Descriptor	9.2.12	All	ONU	280	Yes	Yes
ANI Counters						
GEM Port Performance Monitoring History Data	9.2.6	Depre-cated	OLT	267	Yes	Yes
GEM Port Network CTP Performance Monitoring History Data	9.2.13	All	OLT	341	Yes	Yes
GAL Ethernet Performance Monitoring History Data	9.2.8	All	OLT	276	Reference	Yes
FEC Performance Monitoring History Data	9.2.9	All	OLT	312	Yes	Yes
Enhanced FEC Performance Monitoring History Data	9.2.22	Not G.984	OLT	453	Yes	Yes
Energy Consumption Performance Monitoring History Data	9.2.14	All	ONU	343	Untested	N/A
ANI XG-PON/XGS-PON Counters						
XG-PON TC Performance Monitoring History Data	9.2.15	XG(S)-PON	OLT	344	No	No
XG-PON Downstream Management Performance Monitoring History Data	9.2.16	XG(S)-PON	OLT	345	No	No
XG-PON Upstream Management Performance Monitoring History Data	9.2.17	XG(S)-PON	OLT	346	No	No
Enhanced TC Performance Monitoring History Data	9.2.23	Not G.984	OLT	454	No	No
Layer 2 Forwarding						
MAC Bridge Service Profile	9.3.1	All	OLT	45	Yes	Yes
MAC Bridge Configuration Data	9.3.2	All	ONU	46	Yes	N/A
MAC Bridge Port Configuration Data	9.3.4	All	OLT	47	Yes	Yes
MAC Bridge Port Designation Data	9.3.5	All	ONU	48	Reference	N/A
MAC Bridge Port Filter Table Data	9.3.6	All	ONU	49	Yes	Yes
MAC Bridge Port Filter Preassign Data	9.3.7	All	ONU	79	Yes	Yes
MAC Bridge Port ICMPv6 Process Preassign Table	9.3.33	All	ONU	348	No	N/A
MAC Bridge Port Bridge Table Data	9.3.8	All	ONU	50	Yes	Yes
IEEE 802.1p Mapper Service Profile	9.3.10	All	OLT	130	Yes	Yes
Dot1X Port Extension Package	9.3.14	All	ONU	290	Reference	Yes
Dot1 Rate Limiter	9.3.18	All	OLT	298	Reference	Yes
Layer 2 VLAN Management						
VLAN Tagging Filter Data	9.3.11	All	OLT	84	Yes	Yes
VLAN Tagging Operation Configuration Data	9.3.12	All	OLT	78	No[1)]	N/A

Table 28 Managed Entity Implementation Overview (cont'd)

Managed Entity Name	ITU-T G.988 Ch.	PON Modes	Creation	ME Class	Imple-mentation	PON Adapter
Extended VLAN Tagging Operation Configuration Data	9.3.13	All	OLT	171	Yes	Yes
Layer 2 Counters						
MAC Bridge Performance Monitoring History Data	9.3.3	All	OLT	51	Reference	Yes
MAC Bridge Port Performance Monitoring History Data	9.3.9	All	OLT	52	Reference	Yes
Ethernet Management						
PPTP Ethernet UNI	9.5.1	All	ONU	11	Yes	Yes
Virtual Ethernet Interface Point	9.5.5	All	ONU	329	Yes	Yes
Power over Ethernet Control	9.5.6	All	ONU	349	Reference	No
Ethernet Counters						
Ethernet Frame Performance Monitoring History Data Upstream	9.3.30	All	OLT	322	Yes	Yes
Ethernet Frame Performance Monitoring History Data Downstream	9.3.31	All	OLT	321	Yes	Yes
Ethernet Performance Monitoring History Data	9.5.2	All	OLT	24	Yes	Yes
Ethernet Performance Monitoring History Data 2	9.5.3	All	OLT	89	Reference	Yes
Ethernet Performance Monitoring History Data 3	9.5.4	All	OLT	296	Yes	Yes
Ethernet Frame Extended Performance Monitoring History Data	9.3.32	All	OLT	334	Partly	Yes
Ethernet Frame Extended Performance Monitoring History Data 64-bit	9.3.34	All	OLT	425	Partly	Yes
Layer 3 Multicast						
Multicast Operations Profile	9.3.27	All	OLT	309	Yes	Yes
Multicast Subscriber Config Info	9.3.28	All	OLT	310	Yes	Yes
Multicast Subscriber Monitor	9.3.29	All	OLT	311	Yes	Yes
Layer 3 Management						
IP Host Config Data	9.4.1	All	ONU	134	Yes	Yes
IPv6 Host Config Data	9.4.5	All	ONU	347	Reference	Yes
IP Host Performance Monitoring History Data	9.4.2	All	ONU	135	No	Yes
TCP/UDP Config Data	9.4.3	All	OLT	136	Reference	N/A
TCP/UDP Performance Monitoring History Data	9.4.4	All	OLT	342	No	N/A
System OAM Management						
Dot1ag Maintenance Domain	9.3.19	All	OLT	299	Reference	No
Dot1ag Maintenance Association	9.3.20	All	OLT	300	Reference	No

Table 28 Managed Entity Implementation Overview (cont'd)

Managed Entity Name	ITU-T G.988 Ch.	PON Modes	Creation	ME Class	Imple-mentation	PON Adapter
Dot1ag Default MD Level	9.3.21	All	ONU	301	Reference	No
Dot1ag MEP	9.3.22	All	OLT	302	Reference	No
Dot1ag MEP Status	9.3.23	All	OLT	303	Reference	No
Dot1ag MEP CCM Database	9.3.24	All	OLT	304	Reference	No
Dot1ag CFM Stack	9.3.25	All	ONU	305	Reference	No
Dot1ag Chassis-Management Info	9.3.26	All	ONU	306	Reference	No
General Purpose						
UNI-G	9.12.1	All	ONU	264	Yes	Yes
OLT-G	9.12.2	All	ONU	131	Yes	Yes
Network Address	9.12.3	All	OLT/ ONU	137	Reference	N/A
Authentication Security Method	9.12.4	All	OLT	148	Reference	N/A
Large String	9.12.5	All	OLT	157	Yes	N/A
OMCI	9.12.8	All	ONU	287	Yes	N/A
Attribute	9.12.10	All	ONU	289	Yes	N/A
General Purpose Buffer	9.12.12	All	OLT	308	No	N/A
Generic Status Portal	9.12.14	All	OLT	330	Reference	N/A
Octet String	9.12.11	All	OLT	307	No	N/A
Openflow Configuration Data	9.12.18	All	OLT	439	No	N/A
Threshold Data 1	9.12.6	All	OLT	273	Yes	N/A
Threshold Data 2	9.12.7	All	OLT	274	Yes	N/A
Threshold Data 64-bit	9.12.17	All	OLT	426	Yes	N/A
Managed Entity	9.12.9	All	ONU	288	Yes	N/A
File Transfer Controller	9.12.13	All	ONU	318	No	N/A
Time Status Message	9.12.19	All	ONU	440	No	N/A
Miscellaneous Management						
PPTP LCT UNI	9.13.3	All	ONU	83	Yes	N/A
Enhanced Security Control	9.13.11	All	ONU	332	Yes	Yes
TWDM PON Management – NG-PON2						
TWDM System Profile	9.16.1	NG-PON2	ONU	442	Yes	Yes
TWDM Channel	9.16.2	NG-PON2	ONU	443	Yes	Yes
TWDM PON Counters – NG-PON2						
TWDM Channel PHY/LODS Performance Monitoring History Data	9.16.3	NG-PON2	OLT	444	Yes	Yes
TWDM Channel XGEM Performance Monitoring History Data	9.16.4	NG-PON2	OLT	445	Yes	Yes

Table 28 Managed Entity Implementation Overview (cont'd)

Managed Entity Name	ITU-T G.988 Ch.	PON Modes	Creation	ME Class	Imple-mentation	PON Adapter
TWDM Channel PLOAM Performance Monitoring History Data Part 1	9.16.5	NG-PON2	OLT	446	Yes	Yes
TWDM Channel PLOAM Performance Monitoring History Data Part 2	9.16.6	NG-PON2	OLT	447	Yes	Yes
TWDM Channel PLOAM Performance Monitoring History Data Part 3	9.16.7	NG-PON2	OLT	448	Yes	Yes
TWDM Channel Tuning Performance Monitoring History Data Part 1	9.16.8	NG-PON2	OLT	449	Yes	Yes
TWDM Channel Tuning Performance Monitoring History Data Part 2	9.16.9	NG-PON2	OLT	450	Yes	Yes
TWDM Channel Tuning Performance Monitoring History Data Part 3	9.16.10	NG-PON2	OLT	451	Yes	Yes
TWDM Channel OMCI Performance Monitoring History Data	9.16.11	NG-PON2	OLT	452	Yes	Yes
Vendor-specific Functions – Verizon OpenOMCI						
Verizon OpenOMCI	–	NG-PON2	ONU	65400	Yes	No
Watchdog Configuration Data	–	NG-PON2	ONU	65403	Reference	N/A
Watchdog PMHD	–	NG-PON2	OLT	65404	Reference	N/A
Flexible Configuration Status Portal	–	NG-PON2	ONU or OLT	65420	Reference	N/A
Flexible Configuration Status Portal PMHD	–	NG-PON2	OLT	65421	Reference	N/A
MAC Swap Loop Configuration	–	NG-PON2	ONU	65425	Reference	N/A
MAC Swap Loop Monitor	–	NG-PON2		65428	Reference	N/A
Extended Remote Debug	–	NG-PON2	ONU	65426	Reference	N/A
SIP UNI Application Server Alarm Status	–	NG-PON2	ONU	65405	No	N/A
POTS UNI Extension	–	NG-PON2	ONU	65415	No	N/A
VoIP Call Diagnostics Part 1	–	NG-PON2	ONU	65416	No	N/A
VoIP Call Diagnostics Part 2	–	NG-PON2	ONU	65417	No	N/A
VoIP Call Diagnostics Part 3	–	NG-PON2	ONU	65418	No	N/A

1) Deprecated managed entity.

4.4 General Aspects of Managed Entity Handling

This section provides general handling instructions for managed entities.

Managed Entity Class

The managed entity class identifies the type of a managed entity. Depending on the class, the OMCI management information base (MIB) contains a single instance or multiple instances of a managed entity.

Managed Entity Identifier

The managed entity identifier (also known as ME ID) is a value selected by the OLT or the ONU, whichever creates it, at the time a managed entity is created. The ONU must track this number to identify the related managed entity.

Managed Entity Relationship

There are two basic types of managed entity relationship, which are:

- Implicit, given by an identical managed entity ID value
- Explicit, by providing a pointer to the managed entity ID of another managed entity

When the OLT tries to configure a managed entity and references another, yet undefined managed entity, this must be ignored. The pointers to non-existing managed entities must be handled as null pointers.

Message Type Mismatch

When an attempt is made to access a managed entity with an unsupported message type, a negative acknowledgment is sent as the response.

4.5 Management Control Channel

The OMCI management of an ONU is handled over a special management channel transmitted on the fiber interface. This interface is referred to as OMCC (ONU Management and Control Channel).

On the fiber, the management packets are transmitted in their native format within a dedicated GEM port. Within the PON IP hardware module, a packet format conversion is executed to forward or receive these packets in the native Ethernet format.

Figure 47 shows the management control packets are enabled to be received and transmitted as regular Ethernet packets through a Linux network interface (`gem-omci`). The special OMCI frames are encapsulated into Ethernet packets according to **[53]**. The `gem-omci` is a Linux bridge network device. This bridge has the `gem-omci` omci GEM network device and a `dummy0` network device as members. The configuration is done automatically by `netifd` based on the UCI configuration. The configuration file `gem-omci.uci-defaults` for the `gem-omci` is part of the `gpon-omci-onu` feed. The dummy network interface is required to ensure that `gem-omci` is always up. This way, the omci daemon starts listening on the `gem-omci` before the omci GEM port is configured. In a running system, the `gem-omci` configuration is found in `/etc/config/network`.

Figure 47 shows an overview of the involved software, firmware, and hardware components and their interaction.

The packet encapsulation details are mentioned in **Section 4.5.1**.

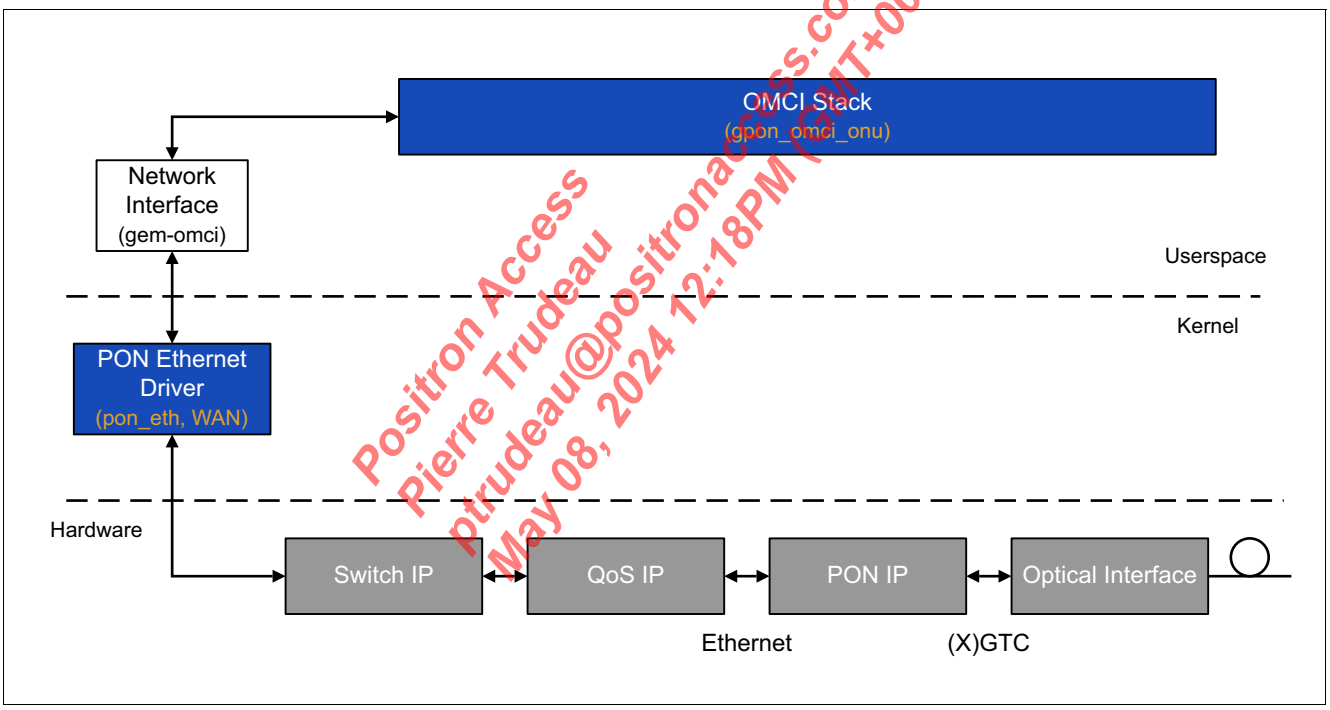
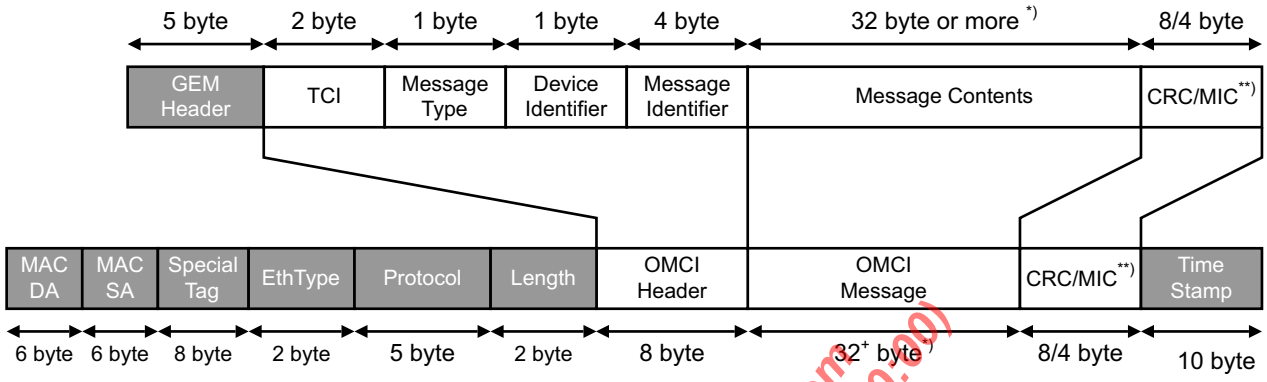


Figure 47 OMCI Management Channel

4.5.1 OMCI Message Encapsulation

The OMCI messages are encapsulated according to **Figure 48** and **Table 29** with a timestamp added to terminate the packet.

The packet header is configured by using the `fapi_pon_omci_cfg_set` function during the device initialization.



*) Using the variable OMCI message sizes may affect the OMCI message length and thus the Ethernet packet size.
**) 8-byte CRC in case of G.984.3, 4-byte MIC in case of G.987.3/989.3/9807.1

Figure 48 OMCI Message Encapsulation into Ethernet Frames

Table 29 OMCI Ethernet Frame Format

Field	Length [byte]	Definition	Value(s)
Packet Header			
MAC Destination Address	6	MAC address of destination station	Unicast MAC address. In downstream, this is the network interface MAC address used for the OMCC. In upstream, it is the PON IP MAC address.
MAC Source Address	6	MAC address of source station	Unicast address. In upstream, this is the network interface MAC address used for the OMCC. In downstream, it is the PON IP MAC address.

Table 29 OMCI Ethernet Frame Format (cont'd)

Field	Length [byte]	Definition	Value(s)
Special Tag	2	Ethertype	0x88C3
	1	Unused (Traffic Class)	0x00
	1	Unused (Control flags) • Bit 7: Destination Enable • Bit 6. OAM Flag • Bit 5: Forced Learning Disable • Bit 4: Traffic Class Enable • Bit 3: Insertion Flag • Bit 2: Extraction Flag • Bit 1: Reserved, set to 0 • Bit 0: Reserved, set to 0	0x00
	1	Unused (Timestamp and Source/Destination Sub-Interface (MSB))	0x00
	1	GEM port index[1)]	Configurable, by default set to 0x00
	1	Unused (Record ID)	0x00
	1	Unused (Record ID/Logical Port ID)	0x00
Ethertype	2	Ethertype to identify the packet payload as OMCI message field.	0x88B7
Protocol	5	Protocol to identify the packet payload as OMCI message field. Defined by ITU-T G.986.	0x0019A70002
Length	2	Length of the OMCI message (bytes). This does include the OMCI header and does also include the CRC/MIC.	0x000A (=10) or larger
OMCI Message			
Transaction Correlation Identifier	2	Unique message identifier	Variable
Message Type	1	See ITU-T G.988	Variable
Device Identifier	1		
Message Identifier	4		
Message Contents	32 or more		
CRC or MIC	8 or 4	OMCI CRC checksum or MIC value	Variable
Packet Trailer			
Timestamp	10	Send time information	Variable

1) The OMCC uses the first GEM port instance and thus always uses the index value 0.

Special Tag

A special Ethertype tag is inserted between the MAC addresses and OMCI payload to allow identification of the packet type and forwarding the GEM port ID information to and from the software. The special tag is removed by the hardware in the downstream direction and inserted by hardware in the upstream. In the switch IP, the GEM port information is carried in the packet meta data header.

GEM Header Calculation

The GEM header is created by the hardware before sending the OMCI frame in the upstream direction.

The header format depends on the G-PON standard used and is automatically selected.

In the downstream, the GEM header is removed by the hardware and not included in the encapsulated data.

OMCI Trailer Calculation

The trailer is calculated by the software before sending in the upstream direction.

The trailer format depends on the G-PON standard used (either CRC or MIC):

- ITU-T G.984.3 (G-PON): CRC
- ITU-T G.987.3 (XG-PON): MIC
- ITU-T G.989.3 (NG-PON2): MIC
- ITU-T G.9807.1 (XGS-PON): MIC

The CRC is calculated without a key, according to the polynomial, as referenced in ITU-T G.984.3. The other G-PON-modes use a message integrity check (MIC) which not only detects bit errors during transmission but also allows the authentication of the message sender. The OMCI integrity key (OMCI_IK) is used as well-known secret at the OLT and the ONU.

To enable the software to check the received OMCI messages and to generate the MIC upon OMCI message transmission, the PON IP firmware calculates and provides the key to be used (OMCI_IK) through a dedicated PON library function (`fapi_pon_omci_ik_get`).

The software implementation uses the OpenSSL library to implement the MIC algorithm. For details of the MIC calculation, see **Section 4.5.1.1**.

Ethernet Frame Checksum

An Ethernet FCS is **not** used for the internal packet exchange between the PON IP and the internal Ethernet MAC of the device.

Timestamp

The timestamp is removed by hardware in the downstream direction and added by hardware in the upstream. This general timestamp handling is applied to all Ethernet data and used for PTP and OAM purposes. It not related to OMCI.

4.5.1.1 OMCI MIC Calculation

The OMCI message integrity check (MIC) is a cryptographic 4-byte checksum over an OMCI message (excluding the last four bytes, which are reserved for the MIC itself), as defined in ITU-T G.987.3 [54]. See Figure 49.

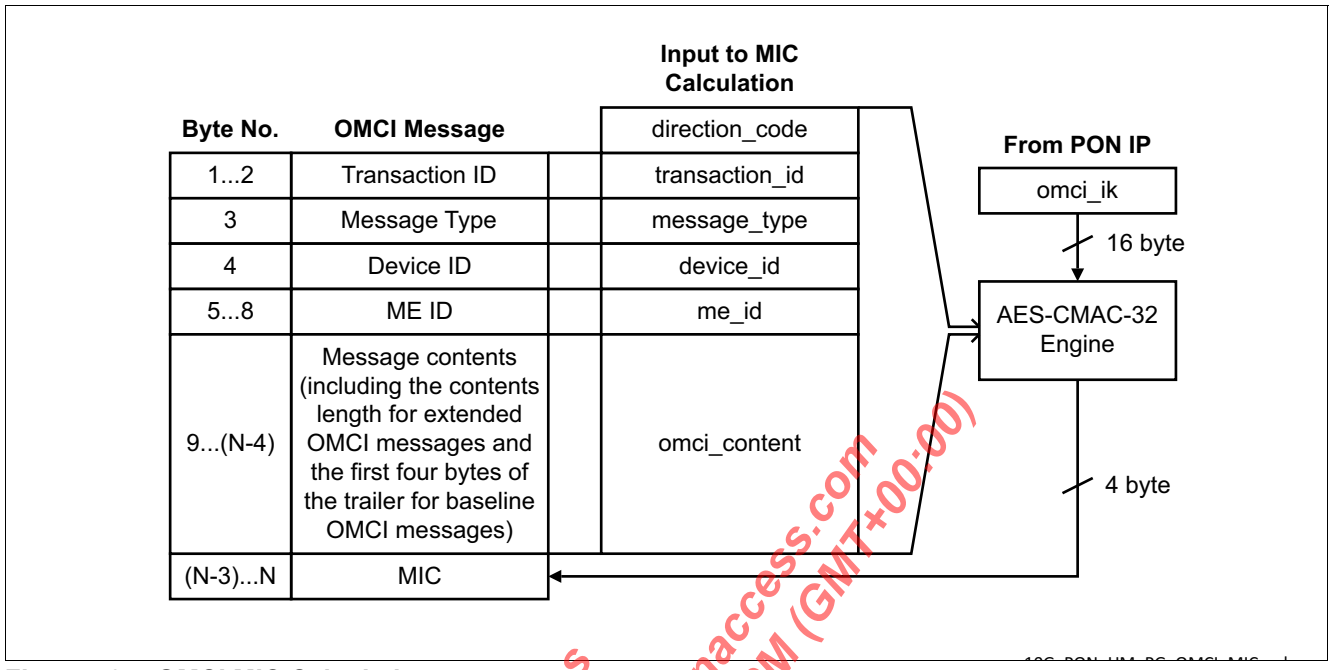


Figure 49 OMCI MIC Calculation

The MIC calculation is defined as:

```
omci_mic = aes_cmac (omci_ik, (direction_code | omci_content), 32)
```

with `direction_code = 0x01` for downstream and `direction_code = 0x02` for upstream OMCI messages. The AES-CMAC algorithm is defined by NIST SP 800-38B [65], using the AES block cypher, as defined in NIST-FIPS-197 [66]. See Figure 50 and Figure 51 for the basic calculation flow.

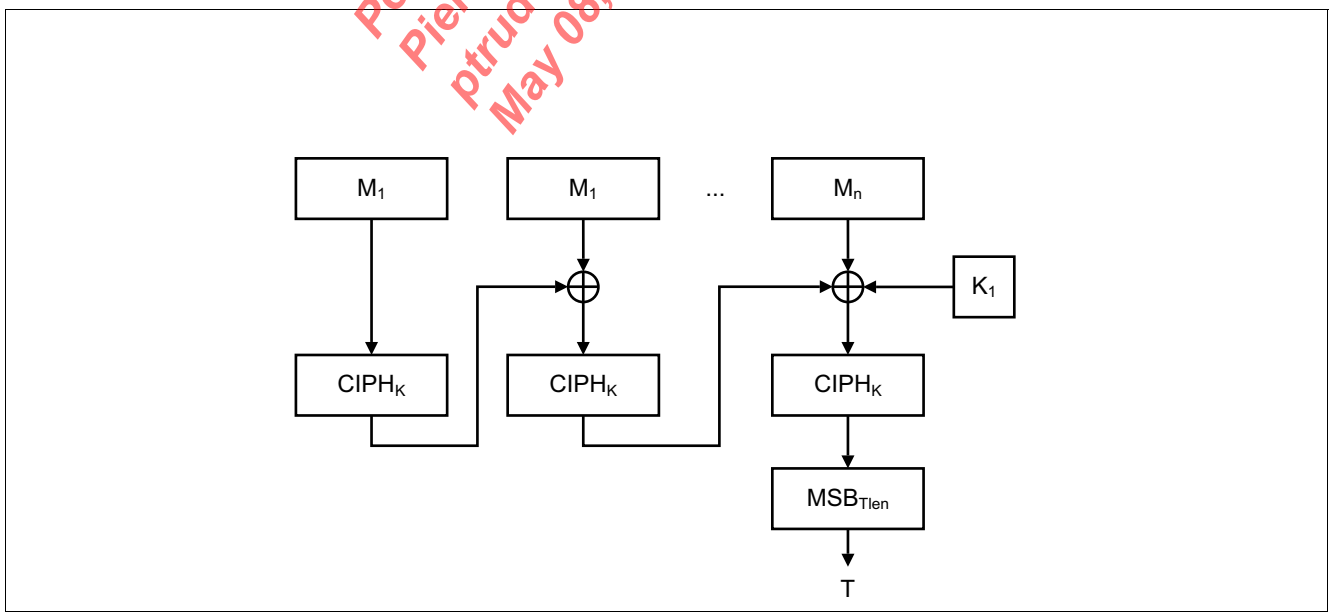


Figure 50 AES-CMAC without Padding

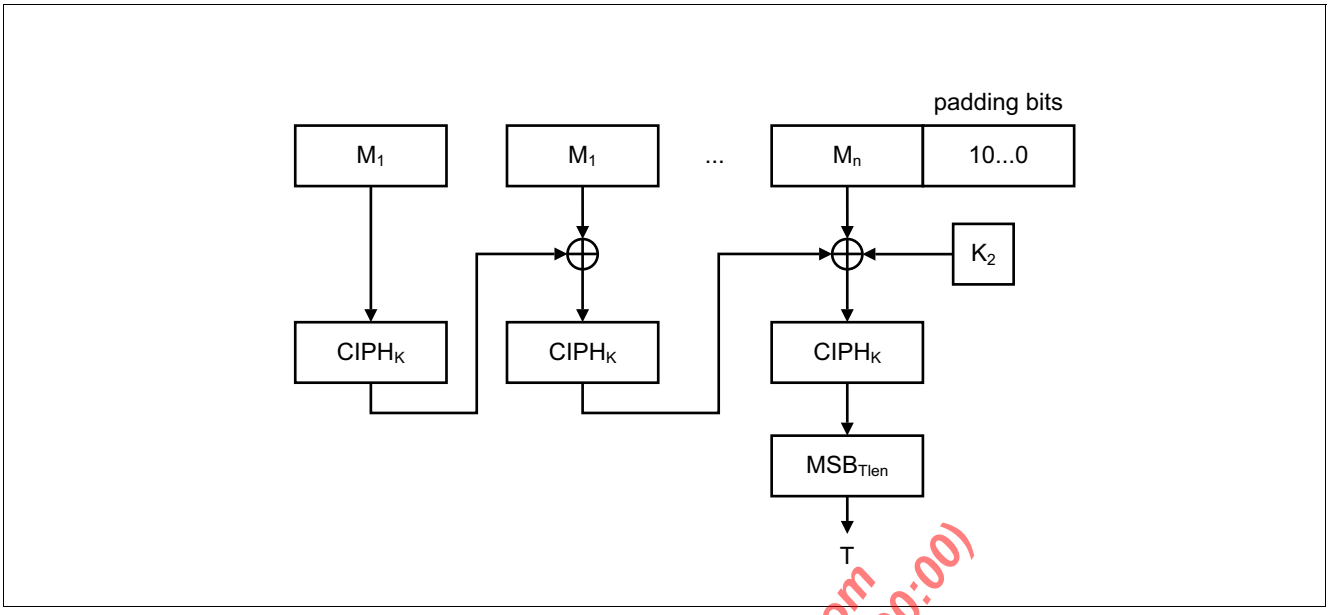


Figure 51 AES-CMAC with Padding

Downstream OMCI MIC Reference

This reference data from ITU-T G.987.3/989.3 ([54], [57]) is used to verify the OMCI MIC implementation:

```
OMCI message direction:
 Cdir = 0x01 (downstream)

OMCI_CONTENT:
 Transaction correlation identifier: 0x80 0x00
 Message type: 0x49 (GET)
 Device identifier: 0x0A (Baseline OMCI)
 Managed entity identifier: 0x01 0x00 (ONU-G)

 Message contents:
     0x00 0x80 0x00 0x00 0x00 0x00 0x00 0x00
     0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
     0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
     0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

OMCI trailer[1:4]: 0x00 0x00 0x00 0x28
OMCI_IK = 0x184b8ad4d1ac4af4dd4b339ecc0d3370

AES-CMAC (OMCI_IK, (Cdir | OMCI_CONTENT), 32)
0x78dca53d
```

OMCI Integrity Key Update

Each time the PLOAM activation state returns to state O5.x, the OMCI_IK is re-generated and read by the OMCI software using the function `fapi_pon_omci_ik_get`.

The PON subsystem generates an event each time the OMCI_IK value is updated (`PON_ALARM_EDGE_OIKC`).

4.6 OMCI Message Handling

This section describes the OMCI message structure and basic usage.

The OMCI messages are received on a Linux network interface, encapsulated in regular Ethernet packets with a special proprietary Ethertype to distinguish them from regular Ethernet data. The encapsulation format is controlled by the PON IP hardware module and is configurable through the PON library. See **Section 4.5** and the PON library documentation for details. The message handler strips the Ethernet encapsulation and provides the raw OMCI message for further processing.

4.6.1 Message Structure

Table 30 shows the baseline message structure details.

Table 30 OMCI Message Structure – Standard Message Format

Field	Number of Bytes	Usage
GEM Header	4	Reduced GEM frame delimiter, containing the port ID
Transaction Correlation Identifier	2	The transaction correlation identifier (TCI) is used to associate a request message with its response message. For request messages, the OLT selects any transaction identifier. A response message carries the transaction identifier of the message to which it is responding. The transaction identifier of event messages is 0000_H. The most significant bit of the transaction correlation identifier indicates the priority of the message. It is up to the OLT to define a message priority, the ONU must use the same priority for the response. Unsolicited event messages must be sent with a value of 0. The OLT must ensure that the currently active TCI values are unique.
Message Type	1	Bit 7 (MSB): Always 0_B in OMCI (Destination Bit, DB) Bit 6: Acknowledge Request (AR) 0_B **NOACK** No acknowledge is required. 1_B **ACK** A response to an action request is required. Bit 5: Acknowledge indication (AK) 0_B **NOACK** This is not an acknowledge message. 1_B **ACK** This is an acknowledge message. Bit 4:0: Message Type (MT). See **Table 32**.
Device Identifier	1	Fixed value of $0A_H$
Message Identifier	4	The message identifier consists of four bytes. The first, most significant, 2 bytes of the message identifier field are used to indicate which managed entity is the target of the action specified in the message type. The least significant two bytes of this message identifier field are used to identify the managed entity instance.
Message Contents	32	Message text, the structure depends on the message type.
OMCI Trailer – Part 1	2	Set to 0000_H for transmission and ignored upon reception.
OMCI Trailer – Part 2	2	Message length, always set to 0028_H.
OMCI Trailer – Part 3	4	32-bit CRC checksum according to ITU-T I.363.5

Table 31 shows the extended messages structure details.

Table 31 OMCI Message Structure – Extended Message Format

Field	Number of Bytes	Usage
GEM Header	4	Reduced GEM frame delimiter, containing the port ID
Transaction Correlation Identifier	2	The transaction correlation identifier (TCI) is used to associate a request message with its response message. For request messages, the OLT selects any transaction identifier. A response message carries the transaction identifier of the message to which it is responding. The transaction identifier of event messages is 0000_H. The most significant bit of the transaction correlation identifier is **not** used to indicate the priority of the message. Unsolicited event messages must be sent with a value of 0. The OLT must ensure that all the currently active TCI values are unique.
Message Type	1	Bit 7 (MSB): Always 0_B in OMCI (Destination Bit, DB) Bit 6: Acknowledge Request (AR) 0_B **NOACK** No acknowledge is required. 1_B **ACK** A response to an action request is required. Bit 5: Acknowledge indication (AK) 0_B **NOACK** This is not an acknowledge message. 1_B **ACK** This is an acknowledge message. Bit 4:0: Message type. See Table 32.
Device Identifier	1	Fixed value of $0B_H$
Message Identifier	4	The message identifier consists of four bytes. The first, most significant, 2 bytes of the message identifier field are used to indicate which managed entity is the target of the action specified in the message type. The least significant two bytes of this message identifier field are used to identify the managed entity instance.
Message Contents Length	2	Length of the message text, given in number of bytes. The minimum length is 0, the maximum length is 1966 byte.
Message Contents	32	Message text, the structure depends on the message type.
Message Integrity Check	4	In ITU-T G.984 systems, this is a 32-bit CRC checksum according to ITU-T I.363.5. In ITU-T G.988 [56] systems, this is a 32-bit message integrity checksum (MIC) according to ITU-T G.987.3.

Extended OMCI messages are only partially supported. The PON interface circuitry drops all *received* OMCI messages smaller than 48 bytes. Therefore short extended OMCI messages must be padded by the OLT. The *transmission* of short messages in upstream direction is not restricted.

4.6.2 Transaction Correlation Identifier

The TCI is used to relate response messages to request through an identical number.

An additional option is to use the MSB for priority control but for baseline messages only. The implementation uses the same MSB setting for responses but does not differentiate the handling of high and low priority OMCI messages.

Note: The OMCI message priority handling must be added by vendor-specific enhancements, when required.

The TCI is also used to protect against replay attacks. The OMCI messages with different contents but identical or smaller TCI are rejected.

This automatic handling can be modified. For example use the CLI function `omci tci_check_ignore` to change the setting for testing:

```
# omci tci_check_ignore
Long Form: tci_check_ignore
Short Form: tcici

Input Parameter
- enum tci_check_mode tci_ignore
   TCI_FULL_CHECK = 0
   TCI_NO_CHECK = 1
   TCI_NO_SEQ_CHECK = 2
```


4.6.3 Message Types

Table 32 lists the available message types.

Table 32 OMCI Message Types

MT[4:0] Value	Type	Purpose	AK	Increment MIB Data Sync Value
1	Reserved	–	–	–
2	Reserved	–	–	–
3	Reserved	–	–	–
4	Create	Creates a managed entity instance with its attributes	Yes	Yes
5	Reserved	–	–	–
6	Delete	Deletes a managed entity instance	Yes	Yes
7	Reserved	–	–	–
8	Set	Sets one or more attributes of a managed entity	Yes	Yes
9	Get	Gets one or more attributes of a managed entity	Yes	Yes
10	Reserved	–	–	–
11	Get all alarms	Latches the alarm statuses of all managed entities and reset the alarm message counter	Yes	No
12	Get all alarms next	Gets the active alarm status of the next managed entity	Yes	No
13	MIB upload	Latches the MIB	Yes	No
14	MIB upload next	Gets latched attributes of a managed entity instance	Yes	No

Table 32 OMCI Message Types (cont'd)

MT[4:0] Value	Type	Purpose	AK	Increment MIB Data Sync Value
15	MIB reset	Clears the MIB and reinitialize it to its default and reset the MIB data sync counter to 0	Yes	No
16	Alarm	Notification of an alarm	No	No
17	Attribute value change	Notification of an autonomous attribute value change	No	No
18	Test	Requests a test on a specific managed entity	Yes	No
19	Start software download	Starts a software download action	Yes	Yes
20	Download section	Downloads a section of a software image	Yes/No	No
21	End software download	Ends a software download action	Yes	Yes
22	Activate software	Activates the downloaded software image	Yes	Yes
23	Commit software	Commits the downloaded software image	Yes	Yes
24	Synchronize time	Synchronizes the time between the OLT and ONU	Yes	No
25	Reboot	Reboots the ONU or circuit pack	Yes	No
26	Get next	Gets the latched attribute values of the managed entity within the current snapshot	Yes	No
27	Test result	Notification of test result that is initiated by Test	No	No
28	Get current data	Gets current counter value associated with one or more attributes of a managed entity	Yes	No
29	Set table	Sets one or more rows of a table (not supported, used in extended OMCI mode only)	–	–
30	Reserved	–	–	–
31	Reserved	–	–	–

4.6.4 Attribute Mask

Each OMCI message is able to access one or more attributes that are part of a managed entity. Up to 16 attributes can be defined and any combination is accessible by an OMCI message. The attributes to be read or written are identified by the attribute mask, where each bit position is related to one of the attributes. The attributes are numbered in the order they are defined for each managed entity.

Table 33 shows the attribute number assignment to the attribute mask bytes.

Table 33 Attribute Mask Definition

Attribute Mask Byte	Bit Position							
	8 (MSB)	7	6	5	4	3	2	1 (LSB)
1	1	2	3	4	5	6	7	8
2	9	10	11	12	13	14	15	16

4.7 OMCI Attribute Value Change Handling

Several managed entities provide Attribute Value Change (AVC) notifications. The handling of these is described in the implementation description of each managed entity. Table 34 provides an overview.

Table 34 OMCI AVC Overview

Managed Entity	AVC Name	AVC ID	Support
ONU-G	Operational State	8	Supported
	LOID	10	This AVC is sent when the LOID value is changed through local access. OEM extension required.
	Logical Password	11	This AVC is sent when the logical password value is changed through local access. OEM extension required.
ONU2-G	OMCC Version	8	Not applicable. The OMCC version does not change during operation.
Software Image	Version	1	Handled by the OMCI software update process.
	Is committed	2	
	Is active	3	
	Is valid	4	
	Product Code	5	
	Image Hash	6	
Cardholder	Actual Type	1	Not applicable, only non-pluggable equipment is supported.
	Actual Equipment ID	5	
	ARC Timer Expiration	8	
Circuit Pack	Operational State	7	Not applicable. Only non-pluggable equipment is supported.
ONU Power Shedding	Shedding Status	11	OEM extension required
ANI-G	ARC	8	Supported
GEM ITP	Operational State	6	Supported
Multicast GEM ITP	Operational State	6	Supported
IP Host Config Data	Current Address	9	OEM extension required
	Current Mask	10	OEM extension required
	Current Gateway	11	OEM extension required
	Current Primary DNS	12	OEM extension required
	Current Secondary DNS	13	OEM extension required
	Domain Name	14	OEM extension required
	Host Name	15	OEM extension required
IPv6 Host Config Data	Current Address Table	9	OEM extension required
	Current Default Router Table	10	OEM extension required
	Current DNS Table	11	OEM extension required
	Current On-link Prefix Table	14	OEM extension required

Table 34 OMCI AVC Overview (cont'd)

Managed Entity	AVC Name	AVC ID	Support
PPTP Ethernet UNI	Sensed Type	2	Supported
	Operational State	6	Supported
	ARC Timer Expiration	12	Supported
Virtual Ethernet Interface Point	Operational State	2	Supported
Enhanced Security Control	ONU Random Challenge Table	5	Supported
	ONU Authentication Result Table	6	Supported
	ONU Authentication Status	9	Supported
Large String	Number of Parts	1	Supported
	Part 1 to 15	2 to 16	Supported

4.8 OMCI Alarm Handling

Several managed entities provide alarm notifications. The handling of those is described in the implementation description of each managed entity. **Table 35** provides an overview.

Table 35 OMCI Alarm Overview

Managed Entity	Alarm Name	Alarm ID	Support
ONU-G	Equipment alarm	0	OEM extension required
	Powering alarm	1	OEM extension required
	Battery missing	2	OEM extension required
	Battery failure	3	OEM extension required
	Battery low	4	OEM extension required
	Physical intrusion	5	OEM extension required
	ONU self test failure	6	OEM extension required
	Dying Gasp	7	Supported via PLOAM only
	Temperature yellow	8	OEM extension required
	Temperature red	9	OEM extension required
	Voltage yellow	10	OEM extension required
	Voltage red	11	OEM extension required
	ONU manual power off	12	OEM extension required
	Invalid software Image	13	Supported
	PSE overload yellow	14	OEM extension required
	PSE overload red	15	OEM extension required
Cardholder	Plug-in circuit missing	0	Not applicable. Only non-pluggable equipment is supported.
	Plug-in type mismatch	1	
	Improper card removal	2	
	Plug-in equipment ID mismatch	3	
	Protection switch	4	
Circuit Pack	Equipment alarm	0	Not applicable. Only non-pluggable equipment is supported.
	Powering alarm	1	
	Self test failure	2	
	Temperature yellow	3	
	Temperature red	4	
Equipment Extension Package	Sense point 1 to 8 alarm	1 to 8	OEM extension required.
ONU3-G	Flash memory performance yellow	0	OEM extension required.
	Flash memory performance red	1	
	Loss of redundant power supply	2	
	Loss of redundant power feed	3	
	Ground fault	4	

Table 35 OMCI Alarm Overview (cont'd)

Managed Entity	Alarm Name	Alarm ID	Support
ANI-G	Low received optical power	0	Supported
	High received optical power	1	Supported
	Signal fail	2	Supported
	Signal degrade	3	
	Low transmit optical power	4	Supported
	High transmit optical power	5	Supported
	Laser bias current	6	Supported
GEM Port Network CTP	End-to-end loss of continuity	5	Supported
Priority Queue	Block loss	0	Not supported
MAC Bridge Port Configuration Data	Port blocking	0	OEM extension required
Dot1X Port Extension Package	Local authentication allowed	0	OEM extension required.
	Local authentication denied	1	OEM extension required.
Dot1ag MEP	RDI CCM	0	OEM extension required.
	MAC status	1	OEM extension required.
	Remote CCM	2	OEM extension required.
	Error CCM	3	OEM extension required.
	Xon CCM	4	OEM extension required.
	Unexpected period	5	OEM extension required.
	AIS	6	OEM extension required.
Multicast Operations Profile	Lost multicast group	0	Not supported
PPTP Ethernet UNI	LAN loss of signal	0	Supported
Virtual Ethernet Interface Point	Connection function fail	0	Not supported

4.9 OMCI TCA Handling

The threshold crossing alerts (TCA) are sent by the OMCI managed entities of the performance monitoring history data (PMHD) type. Depending on the configurable thresholds, the TCA alarm messages are sent by the ONU when a counter value exceeds the selected threshold or falls below the threshold again. The thresholds are configured through dedicated management entities referenced by the individual PMHD managed entities.

Table 36 provides a list of PMHD managed entities and their TCA-capable counters used for NG-PON2 in addition.

Table 36 OMCI TCA Overview – NG-PON2

Managed Entity	TCA Name	Alarm ID	Support
TWDM PON			
TWDM Channel PHY/LODS PMHD	BIP-32 error count	1	
	PSBd HEC errors corrected	2	
	PSBd HEC errors uncorrectable	3	
	FS header errors - corrected	4	
	FS header errors - uncorrectable	5	
	Total LODS event count	6	
	LODS - restored in operating TWDM channel	7	
	LODS - restored in protection TWDM channel	8	
	LODS - restored in discretionary TWDM channel	9	
	LODS - reactivations	10	
	LODS - Handshake failure in protection channel	11	
	LODS - Handshake failure in discretionary channel	12	
TWDM Channel XGEM PMHD	Total transmitted XGEM frames	0	
	Transmitted XGEM frames with LF == 0	1	
	Total Received XGEM frames	2	
	Received XGEM header HEC errors	3	
	FS words lost due to XGEM header HEC errors	4	
	XGEM encryption key errors	5	
TWDM Channel PLOAM PMHD Part 1	PLOAM MIC errors	0	
	Downstream PLOAM message count	1	
	Ranging_time message count	2	
	Protection_Control message count	3	
	Adjust_Tx_Wavelength message count	4	
	Adjust_Tx_Wavelength amplitude	5	

Table 36 OMCI TCA Overview – NG-PON2 (cont'd)

Managed Entity	TCA Name	Alarm ID	Support
TWDM Channel PLOAM PMHD Part 2	System_Profile message count	0	
	Channel_Profile message count	1	
	Burst_Profile message count	2	
	Assign_ONU-ID message count	3	
	Unsatisfied Adjust_Tx_Wavelength requests	4	
	Deactivate_ONU-ID message count	5	
	Disable_Serial_Number message count	6	
	Request_Registration message count	7	
	Assign_Alloc-ID message count	8	
	Key_Control message count	9	
	Sleep_Allow message co	10	
	Tuning_Control/Request message count	11	
	Tuning_Control/Complete_d message count	12	
	Calibration_Request message count	13	
TWDM Channel PLOAM PMHD Part 3	Upstream PLOAM message count	0	
	Serial_Number_ONU (in-band) message count	1	
	Serial_Number_ONU (AMCC) message count	2	
	Registration message count	3	
	Acknowledgment message count	4	
	Sleep_Request message count	5	
	Tuning_Response (ACK/NACK) message count	6	
	Tuning_Response (Complete_u/Rollback) message count	7	
	Power_Consumption_Report message count	8	
	Change_Power_Level parameter error count	9	
TWDM Channel Tuning PMHD Part 1	Tuning control requests for Rx only or Rx & Tx	0	
	Tuning control requests for Tx only	1	
	Tuning control requests rejected/INT_SPC	2	
	Tuning control requests rejected/DS_xxx	3	
	Tuning control requests rejected/US_xxx	4	
	Tuning control requests ok/Target channel	5	
	Tuning control requests failed/TO4 exp.ired	6	
	Tuning control requests failed/TO5 expired	7	
	Tuning control requests resolved/discretionary channel	8	
	Tuning control requests Rollback/COM_DS	9	
	Tuning control requests Rollback/DS_xxx	10	
	Tuning control requests Rollback/US_xxx	11	
	Tuning control requests failed/Reactivation	12	

Table 36 OMCI TCA Overview – NG-PON2 (cont'd)

Managed Entity	TCA Name	Alarm ID	Support
TWDM Channel Tuning PMHD Part 2	Tuning control requests rejected/DS_ALBL	0	
	Tuning control requests rejected/DS_VOID	1	
	Tuning control requests rejected/DS_PART	2	
	Tuning control requests rejected/DS_TUNR	3	
	Tuning control requests rejected/DS_LNRT	4	
	Tuning control requests rejected/DS_LNCD	5	
	Tuning control requests rejected/US_ALBL	6	
	Tuning control requests rejected/US_VOID	7	
	Tuning control requests rejected/US_TUNR	8	
	Tuning control requests rejected/US_CLBR	9	
	Tuning control requests rejected/US_LKTP	10	
	Tuning control requests rejected/US_LNRT	11	
	Tuning control requests rejected/US_LNCD	12	
TWDM Channel Tuning PMHD Part 3	Tuning control requests Rollback/DS_ALBL	0	
	Tuning control requests Rollback/DS_LKTP	1	
	Tuning control requests Rollback/US_ALBL	2	
	Tuning control requests Rollback/US_VOID	3	
	Tuning control requests Rollback/US_TUNR	4	
	Tuning control requests Rollback/US_LKTP	5	
	Tuning control requests Rollback/US_LNRT	6	
	Tuning control requests Rollback/US_LNCD	7	
TWDM Channel OMCI PMHD	OMCI baseline message count	0	
	OMCI extended message count	1	
	OMCI MIC error count	2	

4.10 Unicast Data Flow Configuration

This section shows the internal hardware data flow and the related OMCI configuration instances. **Figure 52** and **Figure 53** provide an overview. **Table 37** provides the relationship between the hardware modules, OMCI managed entities and software components involved in the implementation.

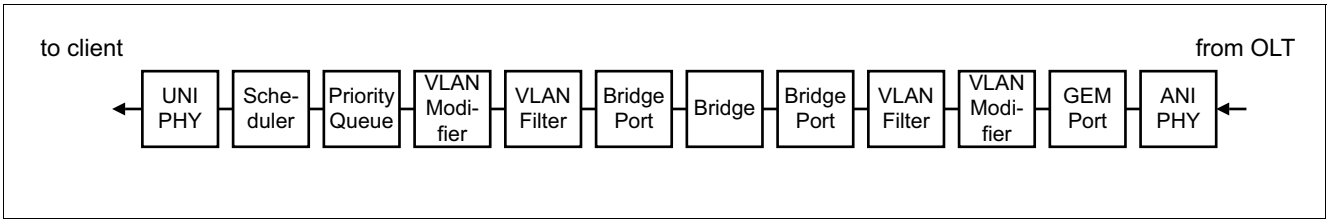


Figure 52 OMCI-controlled Downstream Data Flow

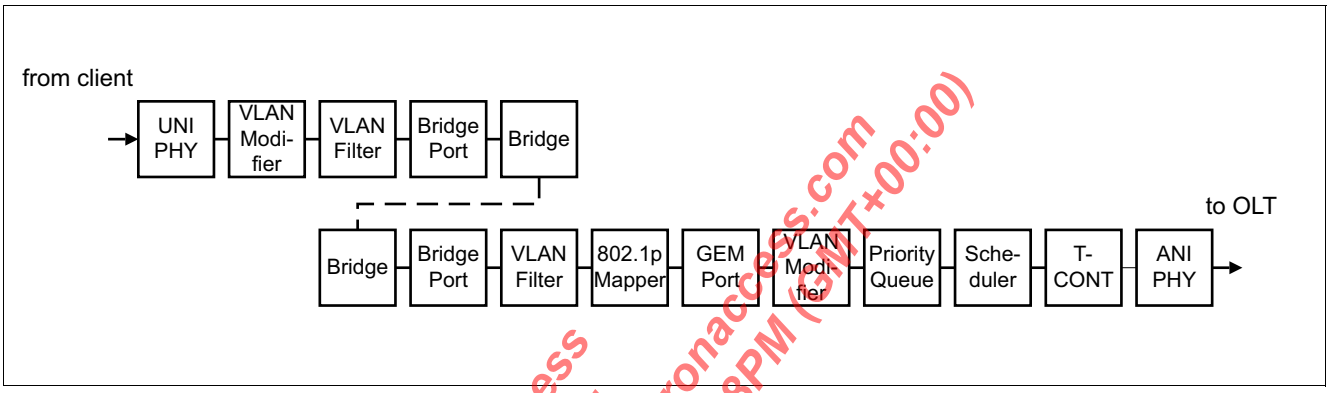


Figure 53 OMCI-controlled Upstream Data Flow

Table 37 OMCI Management Reference – Basic Traffic Flow

Hardware Function	Hardware Module	OMCI Manged Entities	Lower Layer Software Modules
ANI PHY	Optical PMD	ANI-G	PON library
Bridge	Switch IP	MAC Bridge Configuration Data	Linux tc
Bridge	Switch IP	MAC Bridge Service Profile	Linux tc
Bridge Port	Switch IP	MAC Bridge Port Bridge Table Data	Linux tc
Bridge Port	Switch IP	MAC Bridge Port Configuration Data	Linux tc
Bridge Port	Switch IP	MAC Bridge Port Filter Preassign Table	Linux tc
Bridge Port	Switch IP	MAC Bridge Port Filter Table Data	Linux tc
GEM Port	PON IP	GEM Port Network CTP	PON library
GEM Port	PON IP	GEM Interworking TP	PON library
Priority Queue	QoS IP	Priority Queue	QoS driver
Scheduler	QoS IP	Traffic Scheduler	QoS driver
T-CONT	PON IP	T-CONT	PON library
UNI PHY	GPHY	PPTP Ethernet UNI	Linux ethtool

Table 37 OMCI Management Reference – Basic Traffic Flow (cont'd)

Hardware Function	Hardware Module	OMCI Manged Entities	Lower Layer Software Modules
VLAN Filter	Switch IP	VLAN Tagging Filter Data	Data path library
VLAN Modifier	Switch IP	Extended VLAN Tagging Configuration Data	Data path library

4.11 Multicast Data Flow Configuration

This section shows the internal hardware data flow and the related OMCI configuration instances. Figure 54 provides an overview. Table 38 provides the relationship between the hardware modules, OMCI managed entities and software components involved in the implementation.

The multicast access control functions are handled by the `mcc` software module.

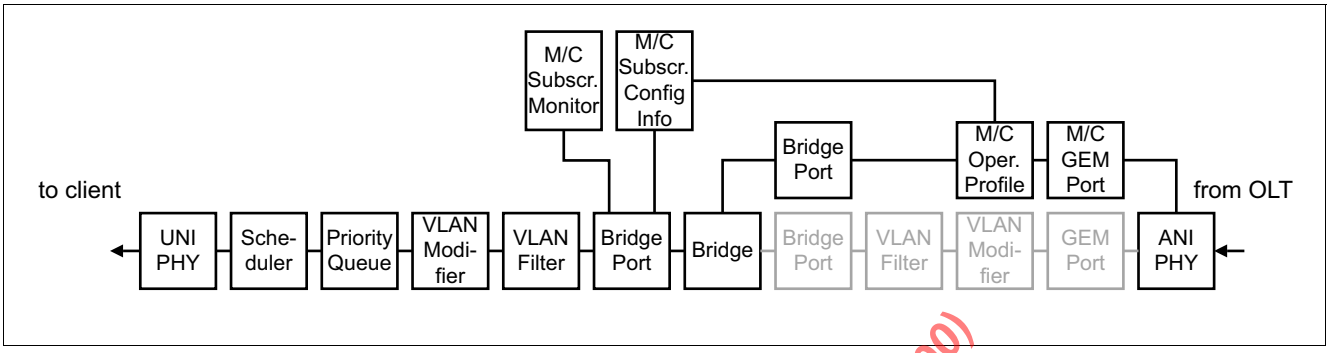


Figure 54 OMCI-controlled Multicast Downstream Data Flow

Table 38 OMCI Management Reference – Multicast Traffic Flow

Hardware Function	Hardware Module	OMCI Manged Entities	Lower Layer Software Modules
ANI PHY	Optical PMD	ANI-G	PON library
Bridge	Switch IP	MAC Bridge Configuration Data	Linux tc
Bridge	Switch IP	MAC Bridge Service Profile	Linux tc
Bridge Port	Switch IP	MAC Bridge Port Bridge Table Data	Linux tc
Bridge Port	Switch IP	MAC Bridge Port Configuration Data	Linux tc
Bridge Port	Switch IP	MAC Bridge Port Filter Preassign Table	Linux tc
Bridge Port	Switch IP	MAC Bridge Port Filter Table Data	Linux tc
GEM Port	PON IP	GEM Port Network CTP	PON library
GEM Port	PON IP	Multicast GEM Interworking TP	PON library
UNI PHY	GPHY	PPTP Ethernet UNI	Linux ethtool
Multicast forwarding	Switch IP	Multicast Operations Profile	Data path library
Multicast VLAN modification	Switch IP	Multicast Operations Profile	Linux tc

4.12 Quality of Service Configuration

This section shows the internal hardware data flow and related OMCI configuration instances related to quality of service. **Figure 55** and **Figure 56** provide an overview. **Table 39** provides the relationship between the hardware modules, OMCI managed entities and software components involved in the implementation

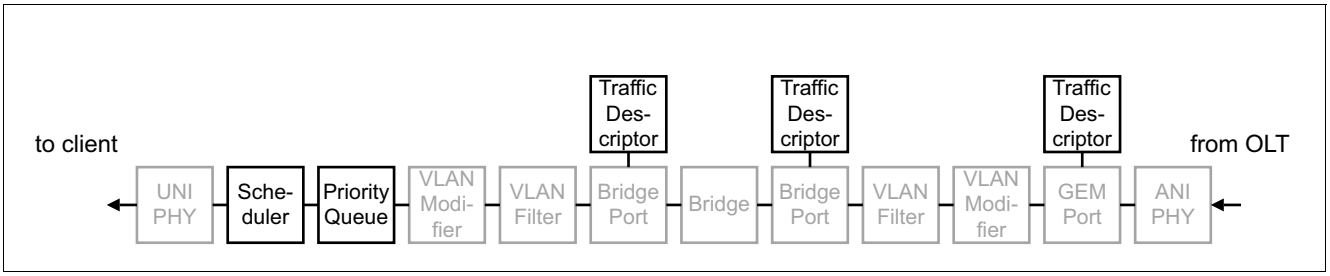


Figure 55 OMCI-controlled Downstream QoS

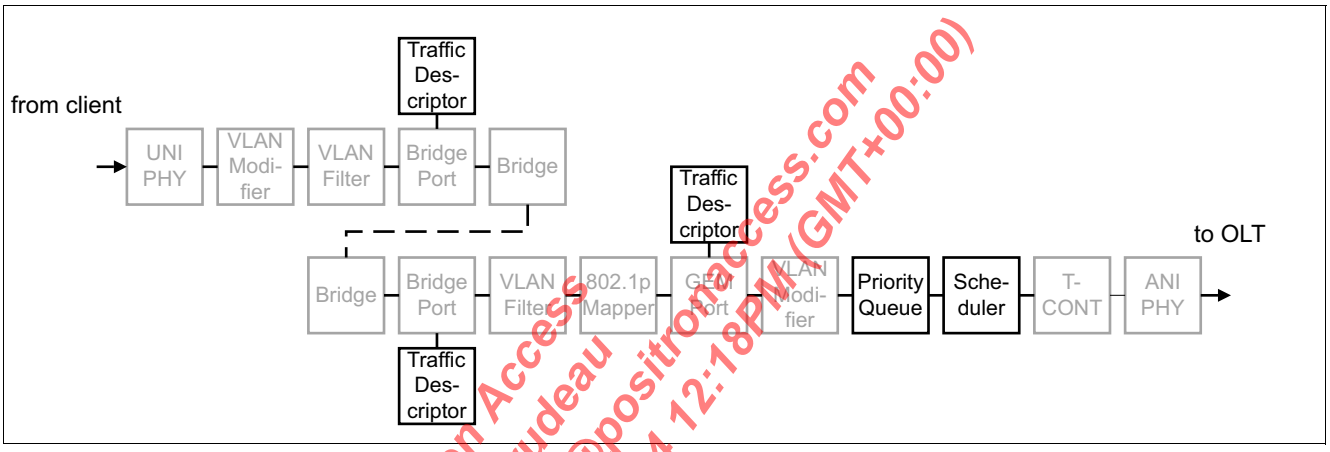


Figure 56 OMCI-controlled Upstream QoS

Table 39 OMCI Management Reference – Quality of Service

Hardware Function	Hardware Module	OMCI Manged Entities	Lower Layer Software Modules
Queue Scheduling Policy	QoS IP	Traffic Scheduler	QoS library
Priority Queue	QoS IP	Priority Queue	QoS library
Traffic Policing at Bridge Port or GEM Port	Switch IP	Traffic Descriptor	Switch API
Traffic Shaping at GEM Port	QoS IP	Traffic Descriptor	QoS library

4.13 Performance Monitoring Configuration

This section shows the internal hardware data flow and OMCI configuration instances related to performance monitoring and counting. **Figure 57** and **Figure 58** provide an overview. **Table 40** provides the relationship between the hardware modules, OMCI managed entities and software components involved in the implementation.

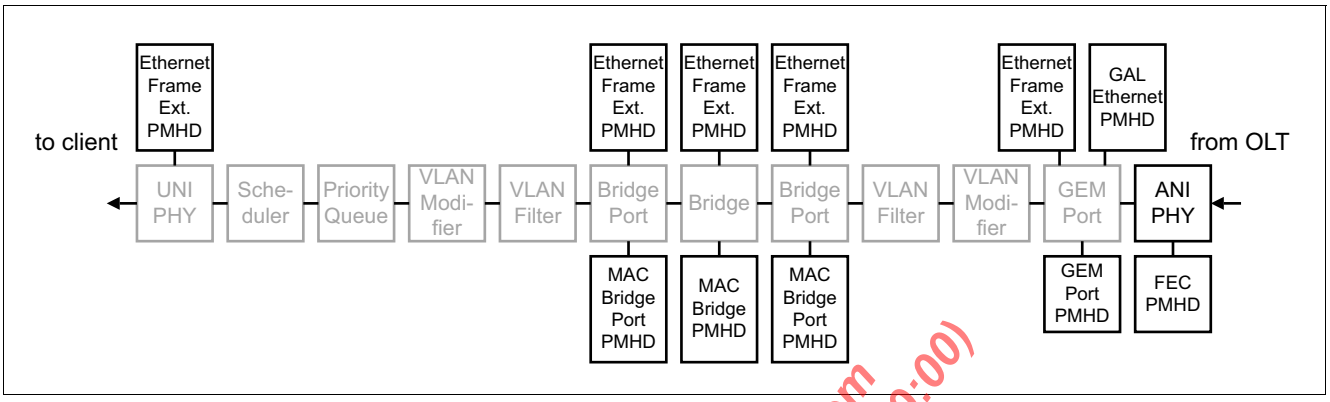


Figure 57 OMCI-controlled Downstream Performance Monitoring

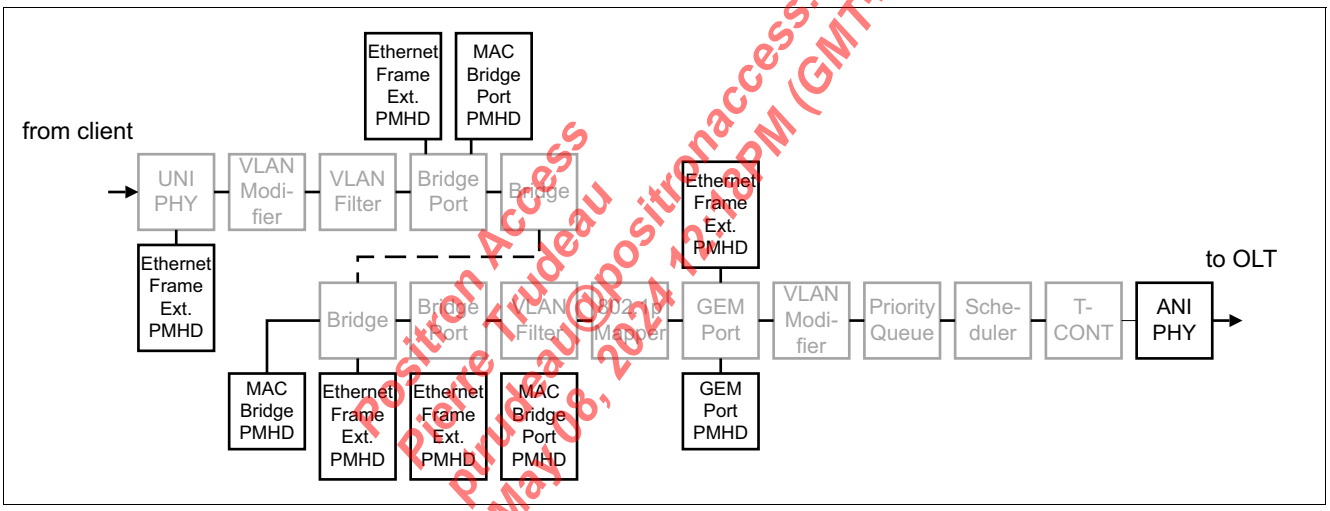


Figure 58 OMCI-controlled Upstream Performance Monitoring

Table 40 OMCI Management Reference – Performance Monitoring

Hardware Function	Hardware Module	OMCI Manged Entities	Lower Layer Software Modules
Ethernet Counting at UNI Port	Switch IP	Ethernet Frame Extended PMHD	Switch API
Ethernet Counting at Bridge Port	Switch IP	MAC Bridge Port PMHD and Ethernet Frame Extended PMHD	Switch API
Ethernet Counting at Bridge	Switch IP	MAC Bridge PMHD and Ethernet Frame Extended PMHD	Switch API
Ethernet Counting at GEM Port	PON IP	GEM Port Network CTP PMHD and Ethernet Frame Extended PMHD	PON library
Optical Supervision	Optical PMD	ANI-G	PON library

4.14 OMCI Daemon Handling

The OMCI daemon `omcid` starts during the system start-up.

See **Section 4.2.4** for details.

4.14.1 MIB Initialization from File

The daemon reads the MIB initialization data from a file containing the latest OMCI configuration successfully applied by the OLT. The MIB data sync attribute is stored as part of the ONU data managed entity. It allows the OLT to check after the MIB upload whether the configuration matches the latest status available at the OLT. When not, the OLT applies a MIB reset and reconfigures the ONU from scratch.

The file to be used is configurable, see **Chapter 2.4.3.7**, **Table 5**.

4.14.2 Default MIB Initialization

When this MIB status file does not exist (for example after a factory reset), a default MIB file is used to provide the initial configuration of all ONU-created managed entities.

4.14.3 Other OMCI Daemon Tasks

Within the OMCI daemon, several other initialization tasks are performed.

During the PON IP initialization, the OMCI daemon:

- Loads the PON IP firmware binary into the hardware module.
- Configures and starts the PON IP function.

4.14.4 MIB Status Information

The actual status of the MIB is stored in two temporary files which are readable by other management systems. One file contains the configuration data and is updated each time the configuration is changed by the OLT. The other file contains the counter values and is updated in regular time intervals.

The file paths are configurable through UCI, see **Chapter 2.4.3.7**, **Table 5**.

The counter storage time interval is defined in the source code as `OMCI_MIB_STATUS_STORE_INTERVAL` in `src/omci_interface.c`. The default value is 10 seconds.

4.15 OMCI Software Structure

The overall software structure and the software layers are shown in **Figure 59**.

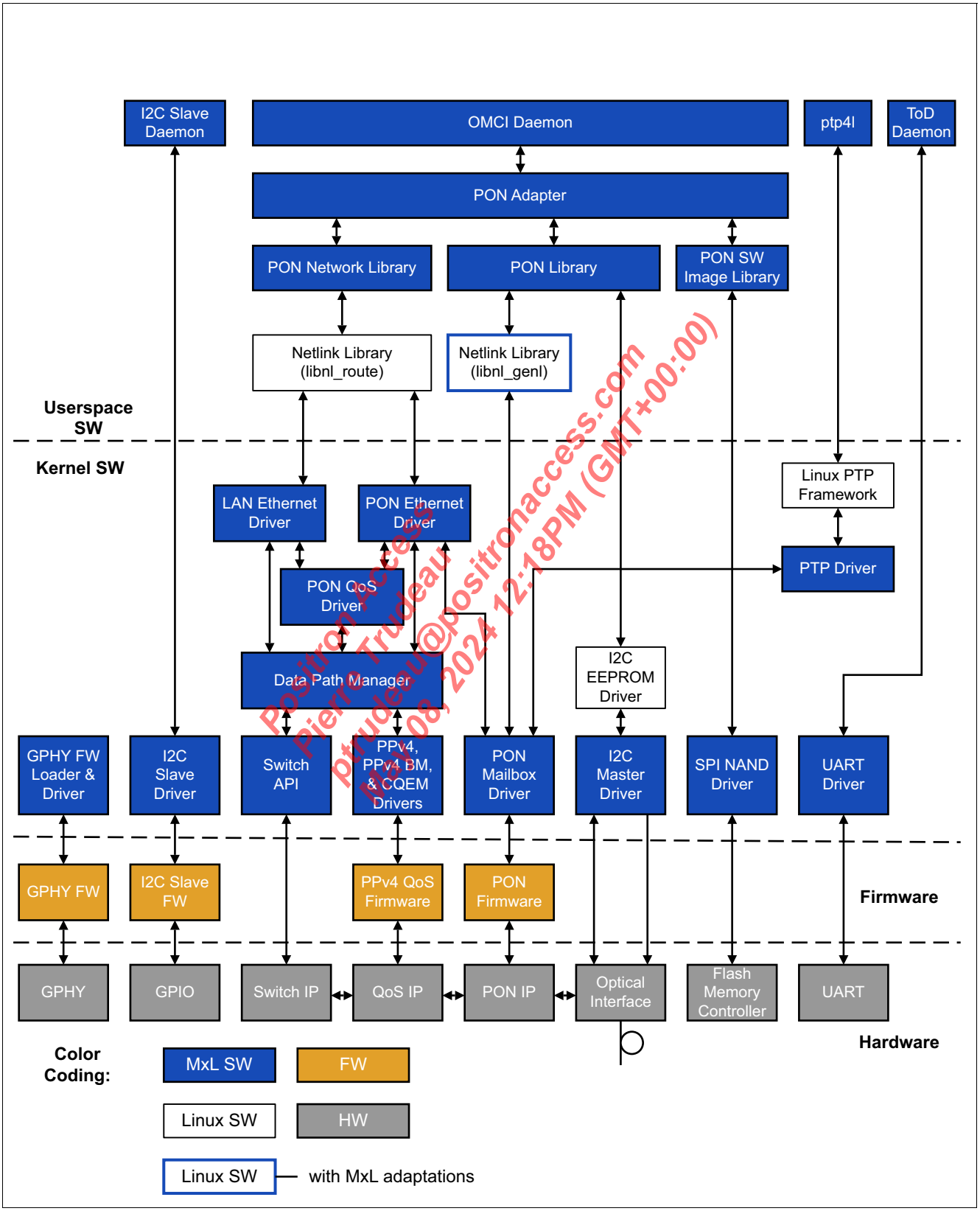


Figure 59 OMCI Software Structure

4.16 OMCI Data Path and QoS Flow in Linux

Figure 60 shows a basic OMCI data path and QoS flow representation in Linux.

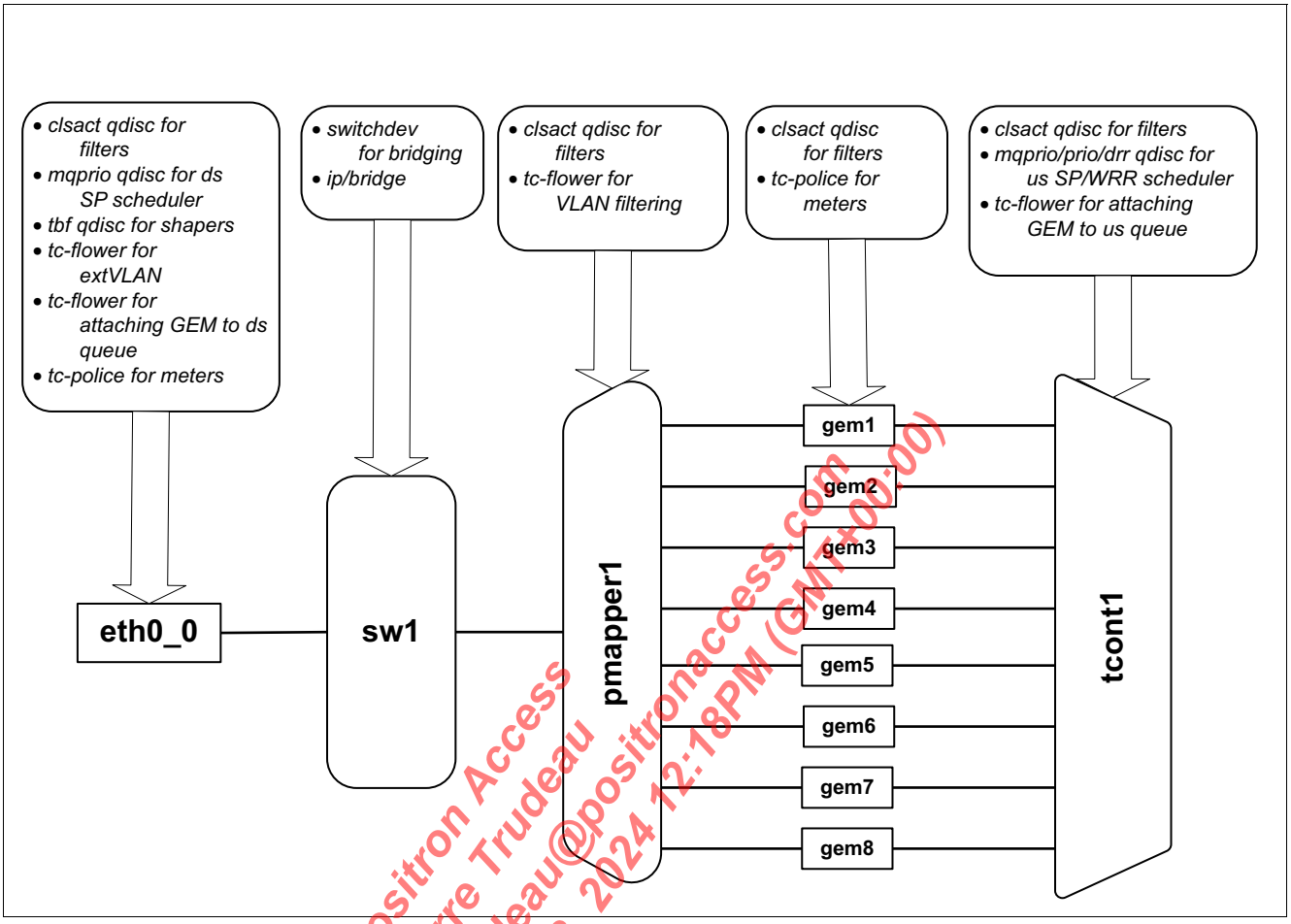


Figure 60 OMCI Data Path and QoS Flow Mapped to Linux Network Devices

The data path flow in **Figure 60** contains theseLinux network devices:

`eth0_0`, `sw1`, `pmapper1`, 8 x `gem`, and `tcont1`.

The connection of these network devices is also shown. The `eth0_0` and `pmapper1` are added to the Linux bridge and the `sw1` becomes their master network device. The rest of the network devices are linked between each other with no special functionality like the Linux bridge. There is a dialog box for each of the network devices describing what kind of qdiscs, filters and actions are used for the configuration of the respective OMCI data path and QoS features. **Table 47** provides more information about the exact mappings.

4.17 OMCI Interaction with PLOAM

The OMCI and PLOAM are two independent management layers, each configuring specific hardware items. However, there is an overlap related to the allocation handling requiring communication between these management layers and aligned interaction. These specifics are described in this section.

The OMCI management is run in software by the OMCI daemon on the main CPU while the PLOAM management is run in firmware within the PON IP hardware module with its dedicated processor.

These managed entities affect the PON IP hardware configuration and the firmware controlling the PLOAM layer:

- **T-CONT**
- **GEM Port Network CTP**
- **GEM Interworking Termination Point**
- **Priority Queue**

While the PLOAM configuration defines which time slot allocation the hardware must respond to, the related OMCI configuration defines which data must be forwarded within these time slots. The connections are made based on the allocation ID, which is configured over PLOAM and OMCI.

As the hardware resources to handle the allocations are limited and the communication over PLOAM is by specification faster than over OMCI, the resource usage must be aligned to avoid misconfiguration.

The number of concurrently supported allocations is read by calling `fapi_pon_cap_get` (`alloc_ids`).

The information about GEM ports and priority queues is only communicated over OMCI, but must also be known by the PON IP module. Thus, this information must also be exchanged in a structured manner.

Hardware and Firmware View

The interaction between the two management layers is organized by a state machine within the PON IP hardware module and its firmware. **Figure 61** shows the state diagram.

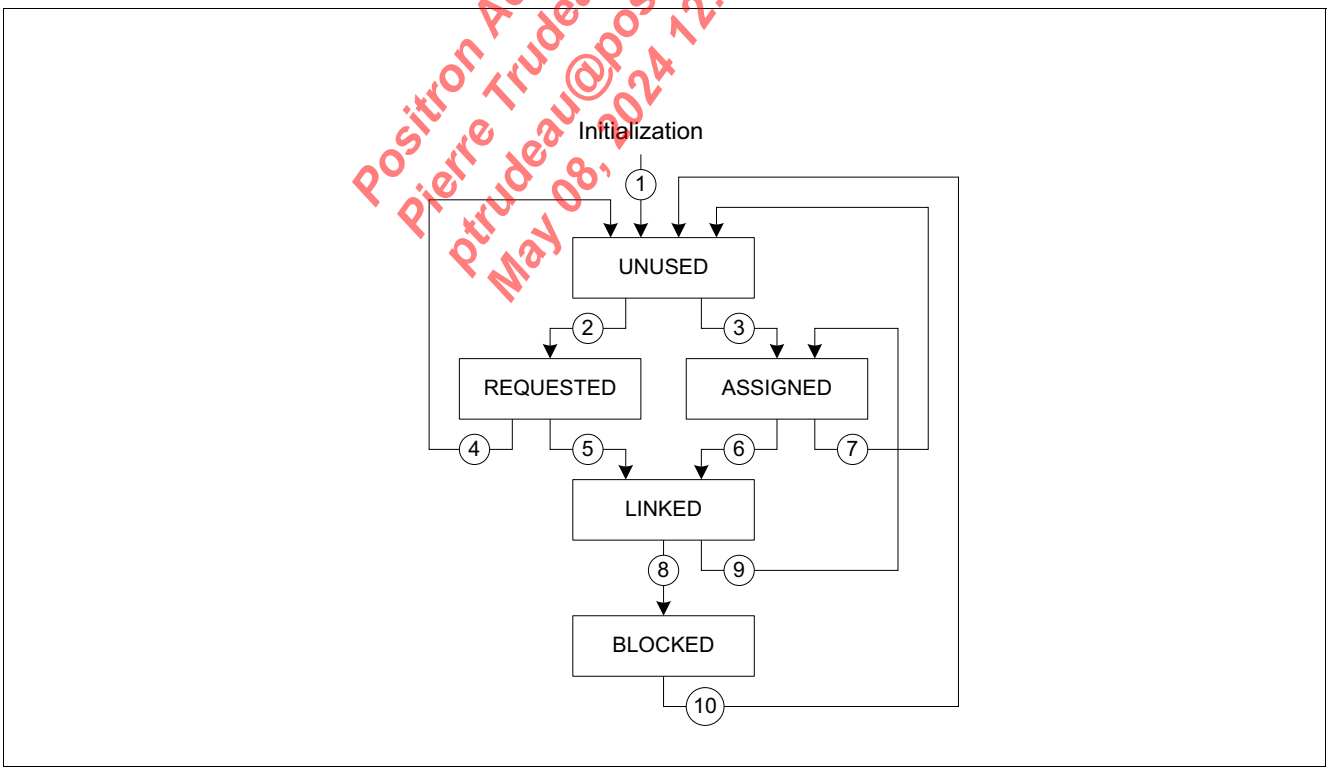


Figure 61 Allocation Hardware and Firmware State Machine

The states are defined per instance of a hardware allocation resource.

- **UNUSED**
  The allocation hardware is not used, neither by PLOAM nor by OMCI. The time slots of this allocation ID are not activated in hardware (not filled with GEM user data nor GEM idle frames).
- **REQUESTED**
  The allocation ID `alloc_id` is assigned by OMCI, but not by PLOAM. The time slots of this allocation ID are not activated in hardware (not filled with GEM idle frames). The OMCI sends the `alloc_id` and `qos_index` information to the firmware and in response receives the `alloc_version` information which must be used for subsequent accesses to this `alloc_id`.
  The `qos_index` identifies the queue scheduler dequeue port from which the data must be sent to the upstream.
- **ASSIGNED**
  The allocation ID is assigned by PLOAM, but not by OMCI. The time slots are activated in hardware and filled with GEM idle frames. No user data is transmitted because the `qos_index` is not known to the PON IP yet.
- **LINKED**
  The allocation ID is known to PLOAM and OMCI with a unique version identifier `alloc_id`, `alloc_version`. The dequeue port `qos_index` is assigned, the GEM ports are assigned by the OMCI. Data is sent in the allocated time slots for all GEM ports `gem_port_id` configured to the PON IP.
- **BLOCKED**
  The allocation ID is unassigned by PLOAM, but not yet by OMCI. The software is informed to make all the queues previously used for this allocation, with their specific version identifier, available again. The time slots of this allocation ID are no longer activated in hardware (not filled with GEM user data or GEM idle frames).

Software View

The software separates the OMCI from the lower hardware and firmware layers. The lower layers are controlled by the configuration information stored in the T-CONT and GEM network devices `netdev` when they are created. The PON Ethernet driver `pon_eth` handles the message communication with the PON hardware and ensures the target configuration as set in the network devices by the OMCI is forwarded to the hardware at the correct point in time.
It also handles the message events from the PON IP. This relieves the OMCI from waiting for hardware events and the firmware from waiting for the OMCI configuration data. **Figure 62** shows the block diagram.

Due to this decoupling, the OMCI stack forwards the configuration anytime it is updated by the OLT without checking for the underlying hardware configuration and PLOAM activation state machine status. It also prevents flooding of the OMCI software by frequent PLOAM message events which may occur during system configuration.

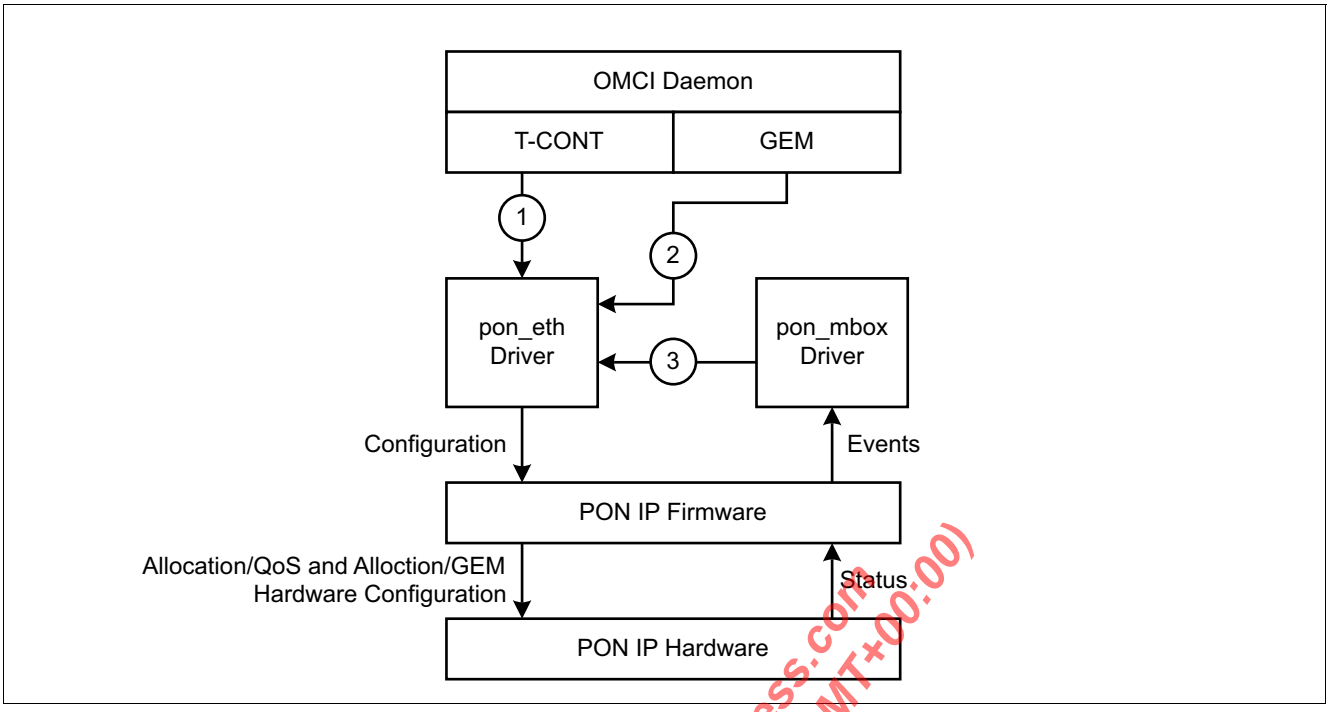


Figure 62 Decoupling OMCI and PLOAM

The major configuration paths are:

1. Allocation ID assignment to T-CONT managed entities
   When a T-CONT receives an allocation ID, the `pon_eth` driver creates the related network device. The driver announces the new allocation ID to the PON IP. The PON IP checks whether this allocation ID is already known over the PLOAM channel and advances the allocation state machine accordingly, either to the REQUESTED or LINKED state (see Figure 61). The allocation information `alloc_id`, `alloc_idx`, `alloc_link_ref`, `qos_idx` is stored in the network device context.
2. Allocation ID de-assignment to T-CONT managed entities
   The network device is removed by the `pon_eth` driver and the driver requests this allocation to be made available again in the PON IP. This moves the used allocation resource into the ASSIGNED state until a PLOAM message de-assigns it on the hardware level and moves it to the UNUSED state.
3. Allocation removal by PLOAM de-assignment
   When an allocation is removed through a PLOAM message before it is removed over OMCI, the allocation resource is moved into the BLOCKED state and an UNLINK event is triggered to the software. The software must acknowledge it to enable the hardware to reuse it again and move it to the UNUSED state.
4. GEM port creation, deletion, or modification
   Upon any GEM port change, a GEM network device is updated: created, modified, or deleted.
   The PON Ethernet driver updates the PON IP hardware configuration to link the GEM port with its allocation ID:
   a) When the allocation is in LINKED state, the GEM port is configured immediately.
   b) When the allocation is in REQUESTED state, the GEM port cannot be configured and the driver waits for a callback.
      Each time the allocation state machine transits from:
      1) REQUESTED to LINKED, or
      2) LINKED to BLOCKED,
      the PON IP sends an event message captured by the PON mailbox driver. This triggers the related callbacks to the PON Ethernet driver which takes the necessary action:
      a) Configures any pending GEM ports to the allocation.
      b) Responds to the allocation removal over PLOAM and frees up the allocation. Then tries to reconfigure the allocation again, unless it has also been removed over OMCI.

4.18 OMCI PON Adapter

The PON adapter layer (`pon_adapter`) is the demarcation between the OMCI stack and the lower layer functionality.

This layer is designed to follow the OMCI managed entity and attribute structure and allows to attach a different OMCI stack software without the need for detailed knowledge of the hardware-specific functionality.

The PON adapter mainly defines the individual data structures that represent the attributes of each managed entity. The data elements of these structures are filled with the related attribute data of the OMCI stack and an update function is called to apply the setting to the underlying software and hardware.

PON Adapter Example: GEM Port Network CTP

This example shows the GEM port configuration. The OMCI code is located in `gpon_omci_onu/src/me/omci_gem_port_network_ctp.c`:

```
static enum omci_error me_update(struct omci_context *context,
                                 struct me *me,
                                 void *data,
                                 uint16_t attr_mask)
{
```

- Define the data structures and pointers:

```
        enum omci_error error = OMCI_SUCCESS;
        struct pa_gem_port_net_ctp_update_data gp_net_ctp_upd_data;
        uint8_t drop_precedence_color_marking = 0, egress_color_marking = 0;
        struct omci_me_gem_port_nw_ctp *upd_data;
        struct me *pq = NULL;
        uint16_t tm_ptr_us = 0x0000,
                 priority_queue_ptr_ds = 0xFFFF;

        dbg_in(__func__, "%p, %p, %p, 0x%04x", context, me, data, attr_mask);

        upd_data = (struct omci_me_gem_port_nw_ctp *)data;

        tm_ptr_us = upd_data->tm_ptr_us;

[...]
```

- Assign the OMCI managed entity attribute values to the PON adapter data structure:

```
        gp_net_ctp_upd_data.gem_port_id = upd_data->gem_port_id;
        gp_net_ctp_upd_data.tcont_ptr = upd_data->tcont_ptr;
        gp_net_ctp_upd_data.direction = upd_data->direction;
        gp_net_ctp_upd_data.us_priority_queue_ptr = tm_ptr_us;
        gp_net_ctp_upd_data.us_priority_queue_drop_precedence_color_marking =
               drop_precedence_color_marking;
        gp_net_ctp_upd_data.us_traffic_descriptor_egress_color_marking =
               egress_color_marking;
        gp_net_ctp_upd_data.us_traffic_descriptor_profile_ptr =
               upd_data->td_ptr_us;
        gp_net_ctp_upd_data.ds_priority_queue_ptr = priority_queue_ptr_ds;
        gp_net_ctp_upd_data.ds_traffic_descriptor_profile_ptr =
```

```
            upd_data->td_ptr_ds;
        gp_net_ctp_upd_data.encryption_key_ring = upd_data->encryption_key_ring;
```

- Call the generic PON adapter macro:

```
        PA_CALL_ALL(context, error,
                    (omci_me_ops, gem_port_net_ctp, update),
                    (me->instance_id, &gp_net_ctp_upd_data));

        dbg_out_ret(__func__, error);
        return error;
}
```

The PON adapter macro functions are defined in `gpon_omci_onu/include/omci_pa.h`.

PON Adapter Example: FEC PMHD

This example shows the readout of the FEC counters. The OMCI code is located in: `gpon_omci_onu_src/me/omci_fec_pmhd.c`

```
static enum omci_error me_counters_get(struct omci_context *context,
                                       struct me *me,
                                       enum omci_pm_interval interval)
{
```

- Define the data structures and pointers

```
        enum omci_error error;
        struct omci_me_fec_pmhd *me_data;
        uint64_t corrected_byte_count = 0,
                 corrected_word_count = 0,
                 uncorrectable_word_count = 0,
                 total_word_count = 0;
        uint16_t fec_seconds = 0;

        dbg_in(__func__, "%p, %p, %u", context, me, interval);

        me_data = (struct omci_me_fec_pmhd *)me->data;
```

- Get the counter values from the hardware through the generic PON adapter macro:

```
        PA_CALL_FIRST(context, error,
                      (omci_me_ops, fec_pmhd, cnt_get),
                      (me->instance_id,
                       &corrected_byte_count,
                       &corrected_word_count,
                       &uncorrectable_word_count,
                       &total_word_count,
                       &fec_seconds));
```

- Assign the counter values received through the PON adapter structure to the OMCI attributes:

```
        /* counters are updated on failure (set to zero)
           and success (set to the values read) */
        me_data->corrected_byte_count = uint64_lo(corrected_byte_count);
        me_data->corrected_word_count = uint64_lo(corrected_word_count);
        me_data->uncorrectable_word_count = uint64_lo(uncorrectable_word_count);
        me_data->total_word_count = uint64_lo(total_word_count);
        me_data->fec_seconds = fec_seconds;

        if (error == PON_ADAPTER_ERR_IF_NOT_FOUND) {
                /* Ignore not existing interface */
                error = PON_ADAPTER_SUCCESS;
        }

        if (error != PON_ADAPTER_SUCCESS) {
                me_dbg_err(me, "DRV ERR(%d) Can't get counters", error);

                dbg_out_ret(__func__, error);
                return error;
        }

        dbg_out_ret(__func__, PON_ADAPTER_SUCCESS);
        return PON_ADAPTER_SUCCESS;
}
```

4.19 OMCI Inter Process Communication

The OMCI daemon communicates with other system processes by using the *ubus* interface [38] as the selected method of Inter Process Communication (IPC). The related open source code can be found at `https://git.openwrt.org/project/ubus.git`.

The OMCI *ubus* implements a data model that is defined by *prplOS* [30]. This data model allows other processes to retrieve OMCI status and configuration information that is requested by another management system. The implementation also allows other processes to read data from the OMCI and register events that are caused by OMCI alarms or the change of data attribute values. A copy of the data model file (xml) can be found on `/gpon_omci/onu/doc`. For details and possible updates of the data model see [32]

The *ubus* interface also includes some TC layer status information beyond the scope of OMCI, for example the PON activation state and state change events.

The *ubus* functionality is optional and must be enabled through the `INCLUDE_OMCI_ONU_UBUS` definition. If enabled, it runs as a separate thread of the OMCI daemon. More details can be found in the source code that is located at `/gpon_omci_onu/src/omci_ubus*`.

The *ubus* interface can be used by any other process that needs to access OMCI data, if that process also implements the *ubus* interface. A command line tool is available for manual debugging, see Chapter 14.14.

4.20 OMCI Managed Entity Implementation

This section describes the programming details for the implementation of dedicated managed entities. It refers to the example implementation which is part of the software delivery. Details may vary for vendor-specific OMCI stack implementations. Vendor-specific managed entity implementations are not covered by this description.

The order of sections follows the order of managed entities in ITU-T G.988 **[56]**, chapter 9. **Table 28** shows an overview listing.

Several managed entities are implemented as empty templates only, to be enhanced with their functionality by the ONU vendor, when required by the target application. The software implementation status of the managed entities and their attributes is stated in the source code. To retrieve the implementation status, the OMCI CLI commands `class_dump_all` and `class_dump_xml` are provided. The generated XML file can be opened by a spreadsheet software for a quick overview.

4.20.1 Equipment Management

These managed entities are defined by ITU-T G.988, in chapter 9.1.

4.20.1.1 ONU-G

ME class: 256

This managed entity is autonomously created by the ONU and does not require hardware programming.

Managed Entity ID

2 bytes, read-only.

The value is fixed to 0.

Vendor Identifier

4 bytes, read-only.

This is stored in a configuration file in an external non-volatile memory.

This value is modifiable in the MIB initialization file.

Version

14 bytes, read-only.

This is stored in a configuration file in an external non-volatile memory.

This value is modifiable in the MIB initialization file.

Serial Number

8 bytes, read-only

This is stored in a configuration file in an external non-volatile memory.

In the 10G PON Chipset System Package, this value is stored in the file `/etc/config/gpon`.

For example:

```
# cat /etc/config/gpon
config 'gpon' 'ploam'
        option 'nSerial' '0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28'
```

The serial number is the same value used within the PLOAM protocol.

To report the serial number in OMCI, use `fapi_pon_credentials_get` to read it back.

*Note: This value **cannot** be changed in the MIB initialization file.*

To allow an easy individualization of ONU boxes, for example in a lab environment, the serial number is automatically created from the MAC address stored in the U-Boot environment (`ethaddr`).

This is done by the script `ugw_sw/openwrt/package/feeds/ugw_pon/pon-lib/files/pond.uci-defaults` upon a first boot, after a software update.

When required, this is customizable to vendor-specific needs.

Traffic Management Option

1 byte, read-only.

This is stored in a configuration file in an external non-volatile memory.

- 0: Priority controlled and flexibly scheduled upstream traffic. The traffic scheduler and priority queue mechanism are used for upstream traffic.
- 1: Rate controlled upstream traffic. The maximum upstream traffic of each individual connection is guaranteed by shaping.
- 2: Priority and rate controlled. The traffic scheduler and priority queue mechanism are used for upstream traffic. The maximum upstream traffic of each individual connection is guaranteed by shaping.

Other values are not supported.

This attribute controls the handling of the GEM/XGEM ports and their relationship to T-CONTs and priority queues. See the **GEM Port Network CTP** attribute **Traffic Management Pointer for Upstream**).

Use the UCI configuration in `/etc/config/omci` (option traffic_management) to modify the value.

Deprecated Attribute

1 byte, read-only.

This value must be set to 0.

Battery Backup

1 byte, read, write.

This is stored in a configuration in an external non-volatile memory.

This value is modifiable in the MIB initialization file.

Administrative State

1 byte, read, write.

When set to 0 (enabled), the ONU operation is allowed. When set to 1 (disabled), all user data traffic is blocked while the OMCI packets are still received and transmitted through the dedicated OMCC channel (Ethernet-encapsulated OMCI messages). This is achieved by configuring a forwarding rule in the GSWIP module to disable all packets except when they are received/transmitted with the OMCC GEM/XGEM port index.

The OMCC GEM/XGEM port index is read by using a PON library function (`fapi_pon_omcc_status_get`) and is set to 0 by default.

Refer to ITU-T G.988 **[56]**, chapter A.1.6 for details.

Operational State

1 byte, read-only.

This attribute reports whether the managed entity is currently capable of performing its function. By default, it is set to 1 (disabled).

These must be provided to set the operational state to 0 (enabled):

- The PON IP firmware is loaded and started.
  The PON capabilities are reported back by using `fapi_pon_cap_get`.

- The optical interface module is accessible through the two-wire (I^2C) interface.

This attribute indicates whether the ONU is able to operate. This is given when the PON firmware is loaded and initialized. This is checked by using `fapi_pon_gpon_status_get` (operational when in PLOAM state O5, ploam_state/10 == 5). When the function returns with an error, the ONU is not operational.

Note: When the ONU is not operational, the OMCI message does not go out to the OLT, but the state can be seen locally when local debugging is enabled.

ONU Survival Time

1 byte, read-only.

This attribute indicates the minimum guaranteed time in milliseconds between the loss of external power and the silence of the ONU. This does not include survival time attributable to a backup battery. The value zero implies that the actual time is not known.

The setting depends on the application hardware architecture, for example the power supply capacitors size. This information is stored in a configuration in an external non-volatile memory and must be provided by the ONU system implementer.

This value is modifiable in the MIB initialization file.

Logical ONU ID

24 bytes, read-only.

This attribute provides a way for the ONU to identify itself. It is a text string, null terminated when it is shorter than 24 bytes, with a null default value. The mechanism for creation or modification of this information is not defined by ITU-T G.988 amd1, but may include for example a web page displayed to a user.
The LOID is modifiable in the UCI configuration (`/etc/config/omci`, see **Section 2.4.3.7**).
This attribute is supported by the software only when the **OMCC Version** is set to 0xA1/0xB1 or later.

Logical Password

12 bytes, read-only.

This attribute provides a way for the ONU to submit authentication credentials. It is a text string, null terminated when it is shorter than 12 bytes, with a null default value. The mechanism for creation or modification of this information is not defined by ITU-T G.988 amd1.

The logical password is modifiable in the UCI configuration (`/etc/config/omci`, see **Section 2.4.3.7**).

Upon system startup, this value is taken from the U-Boot variable `omci_lpasswd`, environment variable when it is present. When this variable does not exist, it is used from UCI. When there is no UCI configuration, it is created from `OMCI_API_DEFAULT_LPASSWD`.

Storing the information in U-Boot allows the setting to be kept during a software update or a factory reset.

This attribute is supported by the software only when the **OMCC Version** is set to 0xA1/0xB1 or later.

Credential Status

1 byte, read, write.

This attribute allows the OLT to signal the ONU whether its credentials are valid. The behavior of the ONU is not specified by ITU-T G.988 amd1. These values are defined:

- 0: Initial state, status unknown
- 1: Successful authentication
- 2: LOID error
- 3: Password error
- 4: Duplicate LOID

Other values are reserved.

This attribute is supported by the software only when the **OMCC Version** is set to 0xA1/0xB1 or later.

Extended TC-Layer Options

2 bytes, read-only.

This attribute provides a bit map which indicates the support of optional PLOAM messages.
This value must be evaluated by calling the `fapi_pon_cap_get` function and checking the bit positions related to `PON_FEATURE_ANXC` and `PON_FEATURE_ANXD`.
This attribute is supported by the software only when the **OMCC Version** is set to 0xA1/0xB1 or later.
This attribute is not supported when the system runs in G.984 operation mode.

Actions

Get, set, reboot, test, synchronize time.

Reboot: After responding to this OMCI request, closes all open files in the file system, shuts down the optical interface and starts a Linux reboot.

Device self test: This is not supported by the device drivers. This attribute is up to vendor-specific implementation.

Synchronize time: This action synchronizes the start time of all PMHD managed entities of the ONU with the reference time of the OLT. All counters of all PMHD MEs are cleared to 0 and restarted. Also, the value of the interval end time attribute of the PMHD MEs is set to 0 and restarted.

Attribute Value Change Indication

These attribute changes are reported:

- AVC #8: Operational state
- AVC #10: LOID
- AVC #11: Logical password

Alarm Notifications

These alarms are defined for this managed entity:

- #0: Equipment alarm (internal interface error)
  - This is specific to the vendor implementation.
    There is no internal interface on which a fault was detected.
- #1: Powering alarm (switching to battery backup)
  - This is for an application that supports external battery backup. The implementation depends on the battery control interface and is implemented through GPIO.
    This function is for vendor-specific implementation and requires a GPIO signal to be configured.
- #2: Battery missing
  - See #1.
- #3: Battery failure
  - See #1.
- #4: Battery low
  - See #1.
- #5: Physical intrusion
  - The implementation is specific to the physical implementation of the ONU.
    A mechanical contact is connected to a GPIO to detect whether the case is opened.
    This function is for vendor-specific implementation and requires a GPIO signal to be configured.
- #6: ONU self test failure
  - When a self test function is implemented, a failure to pass the test must be reported by this alarm.
    This function is for vendor-specific implementation and requires a GPIO signal to be configured.
- #7: Dying Gasp

  - Loss of power in the absence of battery backup or loss of battery power is reported by this alarm. This OMCI alarm must be sent independently of the PLOAM Dying Gasp message handled by the PON IP firmware.
- #8: Temperature yellow
  - This alarm is issued when the yellow temperature threshold is exceeded.
- #9: Temperature red
  - This alarm is issued when the red temperature threshold is exceeded.
- #10: Voltage yellow
  - The implementation is specific to the physical implementation of the ONU. External circuitry is required for supply voltage supervision. This function is for vendor-specific implementation and requires a GPIO signal to be configured.
- #11: Voltage red
  - The implementation is specific to the physical implementation of the ONU. External circuitry is required for supply voltage supervision. This function is for vendor-specific implementation and requires a GPIO signal to be configured.
- #12: ONU manual power off
  - The implementation is specific to the physical implementation of the ONU. A GPIO is used to watch a power button to be pressed and switch the ONU off and on again. The alarm must be issued accordingly. This function is for vendor-specific implementation and requires a GPIO signal to be configured.
- #13: Invalid image
  - The ONU must declare this alarm outside of the software download process, when the downloaded image has been detected to be invalid.
- #14: PSE overload
  - This alarm is applicable only to PoE applications and up to vendor-specific implementation.
- #15: PSE overload red
  - This alarm is applicable only to PoE applications and up to vendor-specific implementation.

4.20.1.2 ONU2-G

ME class: 257

This managed entity is autonomously created by the ONU and does not require hardware programming.

Managed Entity ID

2 bytes, read-only.

The value is fixed to 0.

Equipment Identifier

20 bytes, read-only.

This value is modifiable in the MIB initialization file.

OMCC Version

1 byte, read-only.

This value informs the OLT about the version of OMCC supported by this API.

This is modifiable in the UCI file `/etc/config/omci` and must be given in decimal:

```
config 'omci' 'default'
            option 'mib_file' '/etc/mibs/$M.ini'
            option 'status_file' '/tmp/omci_status'
            option 'omcc_version' '163'
```

The default value used in absence of this UCI configuration must be changed in `/etc/init.d/omcid.sh`.

The versions supported are:

- 0xA0
- 0xA1
- 0xA2,
- 0xA3
- 0xA4

The 10G PON Chipset System Package is delivered with a default version of 0xA3 (163).

Vendor Product Code

2 bytes, read-only.

This value is modifiable in the MIB initialization file.

Security Capability

1 byte, read-only.

This is fixed to 01_H (AES encryption).

This value is modifiable in the MIB initialization file.

Security Mode

1 byte, read-only.

This is fixed to 01_H (AES encryption).

This value is modifiable in the MIB initialization file.

Total Priority Queue Number

2 bytes, read-only.

This is the number of available priority queues (egress queues).

The value is given by the initial MIB configuration and cannot be changed by the OLT.

This value is modifiable in the MIB initialization file.

Total Traffic Scheduler Number

1 byte, read-only.

This is the number of available traffic schedulers.
The value is given by the initial MIB configuration and cannot be changed by the OLT.

This value is modifiable in the MIB initialization file.

Deprecated Attribute

1 byte, read-only.

This is fixed to 01_H.

Total GEM Port ID Number

2 bytes, read-only.

This is the total number of available GEM/XGEM port ID instances. Use the value `gem_ports` through `fapi_pon_cap_get`.

This includes the default GEM/XGEM port ID used to carry the OMCI management channel.

SysUp Time

4 bytes, read-only.

This value counts 10 ms time units since the last system boot (for example, use the Linux uptime command).

Connectivity Capability

2 bytes, read-only.

This attribute reports the supported connectivity model(s). Each bit corresponds to one of the models defined by the OMCI. The value is application-dependent, given by the initial MIB configuration and cannot be changed by the OLT.

The device supports any of the defined connectivity models:

- Bit 0 (LSB):
  - N:1 bridging
- Bit 1:
  - 1:m mapping
- Bit 2:
  - 1:P filtering
- Bit 3:
  - N:M bridge-mapping
- Bit 4:
  - 1:MP map-filtering
- Bit 5:
  - N:P bridge-filtering
- Bit 6:
  - N:MP bridge-map-filtering
- Bit 7 to 15 (MSB):
  - Reserved, set to 0.

Note: The bit numbering in G.988 is from 1 (LSB) to 16 (MSB).

This value is modifiable in the MIB initialization file.

Current Connectivity Mode

1 byte, read, write.

This attribute selects one of the supported connectivity modes.

There is nothing to be configured in hardware. When the OLT tries to select an unsupported model, the setting must be rejected.

QoS Configuration Flexibility

2 bytes, read-only.

This attribute provides multiple 1-bit flags which indicate configuration flexibility of other managed entities:

- **Bit 0** (LSB):
  - 0: The **port** field of the **Related Port** attribute of any **Priority Queue** managed entity cannot be modified and is handled as read-only.
  - 1: The **port** field of the **Related Port** attribute of any **Priority Queue** managed entity can be modified and point to any **T-CONT** or UNI port in the same slot.
- **Bit 1**:
  - 0: The **Traffic Scheduler Pointer – Upstream** attribute of any **Priority Queue** managed entity cannot be modified and is handled as read-only.
  - 1: The **Traffic Scheduler Pointer – Upstream** attribute of any **Priority Queue** managed entity can be modified and point to any other **Traffic Scheduler** in the same slot.
- **Bit 2**:
  - 0: The **T-CONT Pointer** attribute of any **Traffic Scheduler** managed entity cannot be modified and is handled as read-only.
  - 1: The **T-CONT Pointer** attribute of any **Traffic Scheduler** managed entity can be modified to point to any other existing **T-CONT** managed entity.
- **Bit 3**:
  - 0: The **Policy** attribute of any **Traffic Scheduler** managed entity cannot be modified and is handled as read-only.
  - 1: The **Policy** attribute of any **Traffic Scheduler** managed entity can be modified to any of the defined values (0, 1, or 2).
- **Bit 4**:
  - 0: The **Policy** attribute of any **T-CONT** managed entity cannot be modified and is handled as read-only.
  - 1: The **Policy** attribute of any **T-CONT** managed entity can be modified to any of the defined values (0, 1, or 2).
- **Bit 5**:
  - 0: The **priority** field of the **Related Port** attribute of any **Priority Queue** managed entity cannot be modified and is handled as read-only.
  - 1: The **priority** field of the **Related Port** attribute of any **Priority Queue** managed entity can be changed to any valid value.
- **Bit 6 to 15** (MSB): Reserved for future use.

The implementation of this attribute is related to the specific managed entities.

Note: The bit numbering in G.988 is from 1 to 16.

This value is modifiable in the MIB initialization file.

Priority Queue Scale Factor

2 bytes, read, write.

This factor, when provided, is used to scale the queue size and threshold values configured in the **Priority Queue** managed entity. When this attribute is not provided (set to 0), the value given by the **GEM Bock Length** attribute of the **ANI-G** managed entity is used (see **Figure 63**).

When this attribute is provided, set a default value of 1. A value of 0 indicates that this attribute is not provided.

Note: With a `priority_queue_scaling_factor` *of 1, the maximum queue size to be reported and configured is 65536 byte. This is less than the hardware is able to provide. Therefore a default value of 10 is used in the reference software to allow all physical queue memory to be used.*

This factor is also used for dynamic bandwidth reporting in G.984 operation mode and must be configured to the DBR hardware through `pon_pqsf_cfg (psqf)`.

In other operating modes, a fixed block size value of 4 byte is used for DBRu reporting.

This value is modifiable in the MIB initialization file.

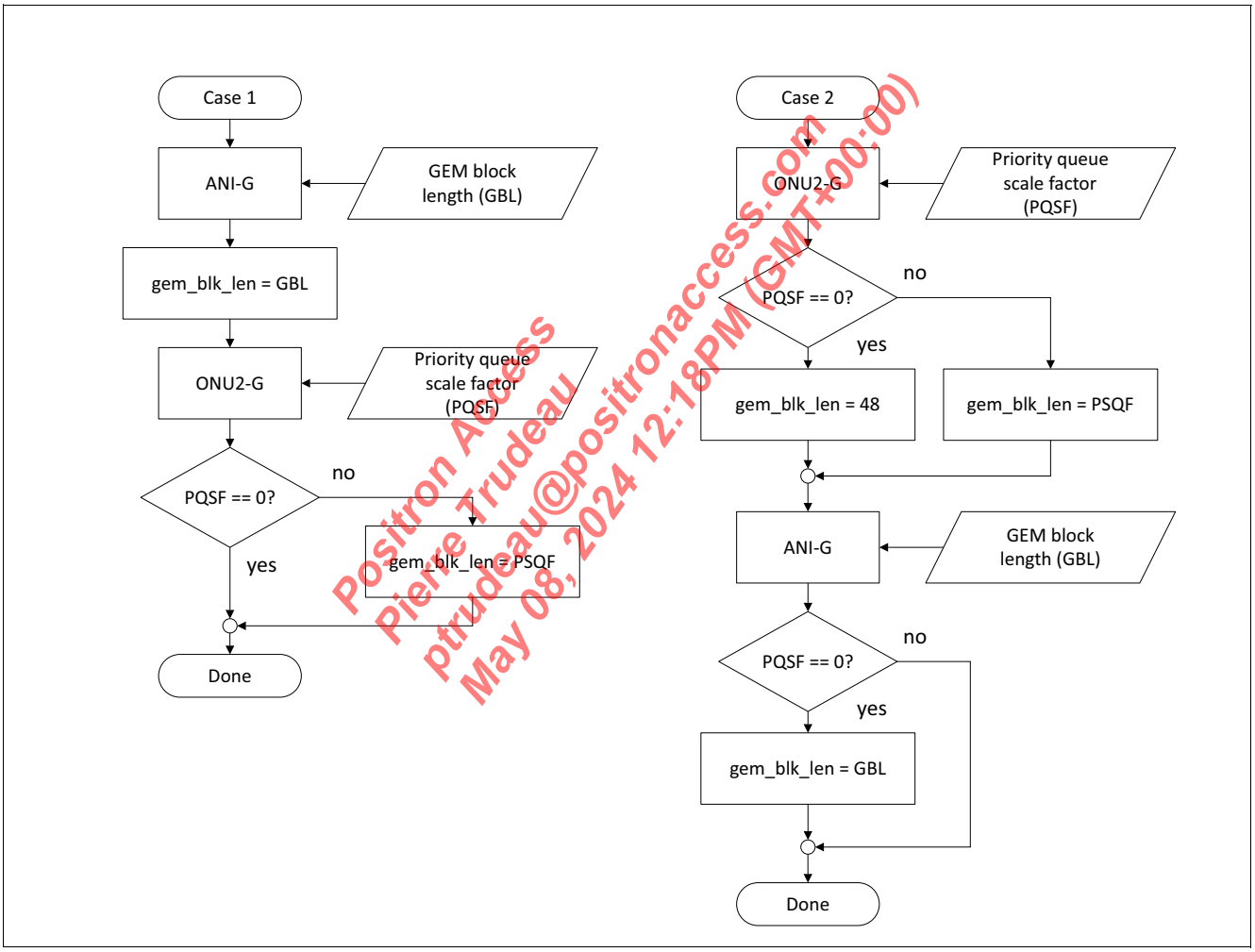


Figure 63 GEM Block Length Configuration Dependencies

Note: A change of this value by the OLT does not change the queue configuration values. The OLT must change the queue configuration values accordingly.

Actions

Get, set

Attribute Value Change Notification

This attribute change is reported:

- AVC #2: OMCC version
  The OMCC version cannot change during operation, as it is only initialized from the configuration file during the OMCI daemon start. Therefore, this AVC is never sent.

4.20.1.3 ONU3-G

ME class: 441

This managed entity is autonomously created by the ONU.

A template implementation is available for reference, the functional implementation is left open for vendor-specific solutions.

Managed Entity ID

2 bytes, read-only.

The value is fixed to 0.

Flash Memory Performance Value

1 byte, read-only

This is a number in the range from 0 to 100 that characterizes the condition of the flash memory, with 0 representing factory fresh device, 100 representing end of life.

The functional implementation is left open for vendor-specific implementation.

Latest Restart Reason

1 byte, read-only

These code points are used to indicate the latest system restart reason:

- 0x00: Unspecified
- 0x01: OLT-initiated software restart remotely by PLOAM or OMCI
- 0x02: User-initiated hardware restart
- 0x03: Self-monitor timer expiration (watchdog timer)
- 0x04: Hardware error (bus time-out, misaligned memory access, or similar a error)
- 0x05: Hardware-triggered automatic restart (voltage monitor or a similar source)
- 0x06: Over-temperature
- 0x07: Software out of memory
- 0x08: Software automatic restart
  (unresolvable references, critical internal inconsistency, or a similar restart reason)
- 0x09: User initiated software restart locally by command line access (CLI) or other user interface
  (such as a WEB GUI).
- 0x0A to 0xDC: Reserved for future use.
- 0xDD to 0xFF: Reserved for vendor-specific use.

Total Number of Status Snapshots

2 bytes, read-only

Maximum size of the snapshot record table (*S*).

The functional implementation is left open for vendor-specific implementation.

Number of Valid Status Snapshots

2 bytes, read-only

Number of valid snapshot records (*M*).

The functional implementation is left open for vendor-specific implementation.

Next Status Snapshot Index

2 bytes, read-only

This attribute identifies the index (ranging from 0 to S - 1) of the next snapshot record to be taken in the snapshot record table.

The functional implementation is left open for vendor-specific implementation.

Status Snapshot Record Table

M x N bytes, read-only

The table of *M* status snapshot records. The size *N* and format of the snapshot record is vendor dependent.

The functional implementation is left open for vendor-specific implementation.

Snap Action

1 byte, write-only

Once the OLT writes this attribute, the ONU takes and records an urgent snapshot without shutting down the transceiver.

The functional implementation is left open for vendor-specific implementation.

Most Recent Status Snapshot

N bytes, read-only

This attribute provides access to the most recently taken status snapshot record.

The functional implementation is left open for vendor-specific implementation.

Reset Action

1 byte, write-only

Once the OLT writes this attribute, the ONU sets the **Number of Valid Status Snapshots** and **Next Status Snapshot Index** attributes to zero.

The functional implementation is left open for vendor-specific implementation.

Enhanced Mode

1 byte, read-only

The boolean value true specifies the “Enhanced Received Frame Classification and Processing Table” attribute is supported by the **Extended VLAN Tagging Operation Configuration Data** managed entity. The value false indicates the “Enhanced Received Frame Classification and Processing Table” is not supported.

Actions

Get, get next, set

Attribute Value Change Notifications

Note: AVC notifications are not handled by the reference implementation and must be added by the system vendor.

AVC#1: **Flash Memory Performance Value** change

AVC#4: **Number of Valid Status Snapshots** change (a new snapshot has been recorded)

Alarm Notifications

Note: The implementation of these alarms is vendor-specific. Alarms are not handled by the reference implementation and must be added by the system vendor.

#0: Flash memory performance yellow (threshold definition is vendor-specific, for example 75)

#1: Flash memory performance red (threshold definition is vendor-specific, for example 90)

#2: Loss of redundant power supply
In an ONU with redundant power supplies, an indication of the loss of one of the two redundant power supplies.

#3: Loss of redundant power feed
In an ONU with dual -48VDC power feeds, an indication of the loss of one of the two power feeds.

#4: Ground fault
Ground fault; ONU has detected a loss of grounding or a degradation in the ground connection.

4.20.1.4 ONU Manufacturing Data

ME class: 456

This managed entity is autonomously created by the ONU.

The data contents of the attributes is defined in the MIB initialization file.
Most of the values are fixed during the lifetime of the ONU.

Some of the attributes require the ONU manufacturer to provide individual values per ONU instance, others may be shared by a series of ONU devices.

Managed Entity ID

2 bytes, read-only.

The value is fixed to 0.

Manufacturer Name

25 bytes, read-only

This attribute contains the manufacturer name of this physical ONU.
The preferred value is the manufacturer name string as it is printed physically on the ONU itself (if present).

Serial Number Part 1

25 bytes, read-only

This is the first of two attributes which together may be regarded as an ASCII string of up to 32 bytes whose length is a left justified manufacturer's serial number for this physical ONU.
The preferred value is the manufacturer serial number string as it is printed physically on the ONU itself (if present).

Serial Number Part 2

25 bytes, read-only

This is the second of two attributes which together may be regarded as an ASCII string of up to 32 bytes whose length is a left justified manufacturer's serial number for this physical ONU.
The preferred value is the manufacturer serial number string as it is printed physically on the ONU itself (if present).

Model Name

25 bytes, read-only

This attribute contains the vendor specific model name identifier string.
The preferred value is the customer-visible part number which may be physically printed on the component itself.

Manufacturing Date

25 bytes, read-only

This attribute contains the date of manufacturing of the physical ONU.
The preferred value is the date of manufacturing as it is physically printed on the ONU itself (if present).

Hardware Revision

25 bytes, read-only

This attribute contains the hardware revision of the physical ONU.
The preferred value is the hardware revision as it is physically printed on the ONU itself (if present).

Firmware Revision

25 bytes, read-only

This attribute contains the vendor-specific firmware revision of the physical ONU.

Actions

Get

Attribute Value Change Indication

No attribute value changes are reported.

Alarm Notifications

No alarms are defined for this managed entity.

4.20.1.5 ONU Time Configuration

ME class: 457

This managed entity is autonomously created by the ONU if OLT-based time synchronization methods are supported.

Note: This managed entity is implemented as a reference structure only. The functionality must be added by the system vendor.

Managed Entity ID

2 bytes, read-only.

The value is fixed to 0.

Current Local ONU Time

7 bytes, read-only

This attribute returns the current ONU time in the same format as defined for the Synchronize Time function of the ONU-G managed entity.

Table 41 Local Time Format

Byte	Description
1 - 2	Year, for example 2023
3	Month, ranging from 1 to 12
4	Day of month, ranging from 1 to 31
5	Hour of day, ranging from 0 to 23
6	Minute of the hour, ranging from 0 to 59
7	Second of the minute, ranging from 0 to 59[1)]

1) A value of 60 may be used if the attribute is read out exactly at the time where a leap second is inserted.

Time Qualification Block

2 bytes, read, write

This attribute describes the time-qualification to be applied to the ONU's local time.

Table 42 Time Qualification Block Coding

Bit	Description
15	Time zone usage 0_B **LOCAL** Use the local time. 1_B **UTC** Use the UTC time.
14	Time zone offset 0_B **NONE** No time zone offset is available. 1_B **AVAIL** The time zone offset is available in the bit fields 10 to 0 according to IETF RFC 3339 **[50]**.
13	Reserved, set to 0.
12	Reserved, set to 0.
11	Time offset sign according to IETF RFC 3339 **[50]**. 0_B **POS** Positive time zone offset. 1_B **NEG** Negative time zone offset.
10:0	Time zone offset given in minutes related to UTC, according to IETF RFC 3339 **[50]**.

Actions

Get, set

Attribute Value Change Indication

No attribute value changes are reported.

Alarm Notifications

No alarms are defined for this managed entity.

4.20.1.6 ONU Operational Performance Monitoring History Data

This managed entity is not implemented. If needed, it must be implemented by the system vendor.

4.20.1.7 ONU4-G

This managed entity is not implemented (it is used for G.HSP applications only).

4.20.1.8 ONU Data

ME class: 2

This managed entity is autonomously created by the ONU and does not require hardware programming.

It is used to handle alarm notifications and to update the MIB data base.

The MIB database status is stored in a file together with the MB data synchronization status and reused upon OMCI start-up to recreate the latest configuration.

The file is stored in a secure encrypted storage and accessed by `ss_fread` and `ss_fwrite` commands.

Managed Entity ID

2 bytes, read-only.

The value is fixed to 0.

MIB Data Synchronization

1 bytes, read, write.

This attribute is used to check the MIB alignment of the ONU with the corresponding MIB in the OLT. The MIB data sync relies on this attribute, which is a sequence number checked by the OLT to see whether the MIB snapshots for the OLT and ONU match. Upon ME instantiation, the ONU sets this attribute to 0.

Actions

- Get
- Set
- Get all alarms
- Get all alarms next
- MIB reset
- MIB upload
- MIB upload next

Attribute Value Change and Alarm Notifications

There are no notifications defined for this managed entity.

4.20.1.9 Software Image

ME class: 7

This managed entity is autonomously created by the ONU and does the required hardware programming for a software image update. The software image is always a complete set, including all flash partitions (when there are more than one) and all firmware images. Individual parts of the software cannot be updated separately.

It stores a new program code in the externally connected non-volatile memory (Flash memory). Two software images are held in parallel to return to a stored image if the update process fails. The handling of the dual-image software update follows the description given in **[56]**, Appendix I.3, Figure I.3.1-1.

For more detail check the software library `pon_img_lib`.

Figure 64 gives a simplified state diagram for reference.

The software image status information is stored in the U-Boot environment to allow to boot from the target image:

- `commit_bank = <A|B>`
- `active_bank = <A|B>`
- `img_validA = <true|false>`
- `img_validB = <true|false>`
- `img_versionA = x.y`
- `img_versionB = x.y`

The image identifier A relates to the Software Image managed entity ID 0 while the identifier B relates to the managed entity ID 1. **Table 43** shows the complete mapping of different variables between the OMCI and U-Boot usage.

Table 43 Software Image Status Variable Mapping

Status	0	1
A0 – Active state of image 0	active_bank = B	active_bank = A
A1 – Active state of image 1	active_bank = A	active_bank = B
C0 – Committed state of image 0	commit_bank = B	commit_bank = A
C1 – Committed state of image 1	commit_bank = A	commit_bank = B
V0 – Valid state of image 0	img_validA = false	img_validA = true
V1 – Valid state of image 1	img_validB = false	img_validB = true

Upon initial ONU software provisioning, the reference implementation flashes the new image into both image partitions. But only `img_validA=true` is set.

If the committed image does not boot because it is broken and another valid image is present, there is no automatic recovery process provided to boot from the other image.

The image selection is done by the U-Boot boot loader.
It is controlled by the environment variable `select_image`.

```
select_image=

if test -n "${img_activate}"; then
        echo "Activated image: ${img_activate}";
        run switchbank${img_activate};
        setenv img_activate; saveenv;

elif test "${commit_bank}" != "${active_bank}"; then
        echo "Committed image: "${commit_bank}"";
        run switchbank"${commit_bank}"; saveenv;
```

```
elif test -n "${commit_bank}";
       then echo "Committed image: ${commit_bank}";
       run switchbank${commit_bank};

else
       echo "no active image"; run switchbankA;

fi;
```

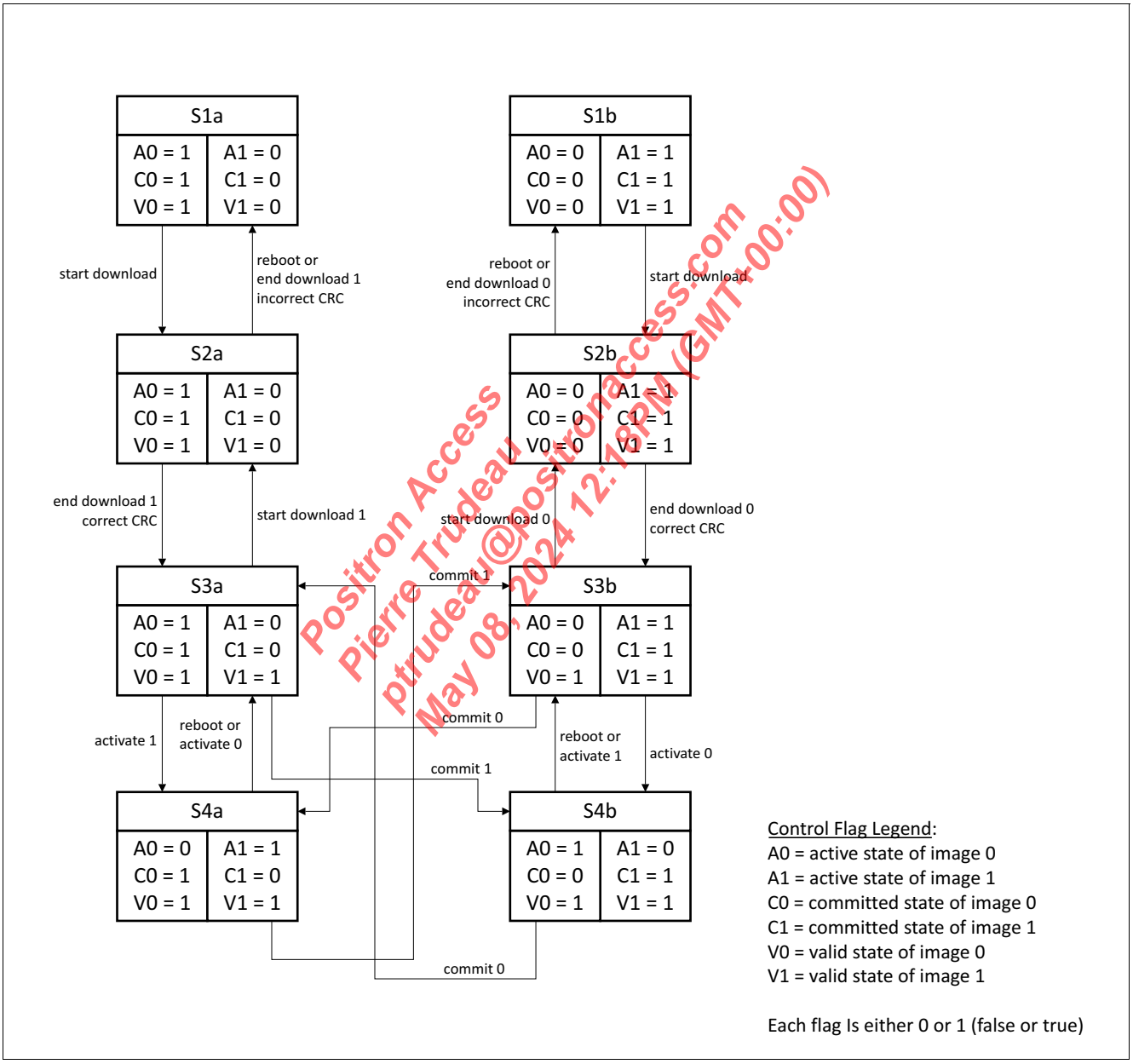


Figure 64 Dual-Image Boot Process

Managed Entity ID

2 bytes, read-only.

This attribute uniquely identifies each instance of this managed entity. The first byte indicates the physical location of the equipment hosting the software image, which is the ONU (0). The second byte distinguishes between the two software image instances (0 or 1).

Version

14 bytes, read-only.

By default, the software version string is stored in a U-Boot environment variable. For each of the two possible images, an individual string is stored in:

- `image0_version`, and
- `image1_version`

The software version string must be defined during the software build process. Use make `menuconfig` and then:

- Select an image configuration
- Select the version configuration options
- Enter a string for the hardware revision

Note: The software version is updated to the U-Boot environment by the OMCI reference implementation. If the software is updated manually from the U-Boot command line (run update_fullimage), this value must be updated manually.

Note: Some OLTs do not accept version identifiers that are larger than 13 bytes.

Is Committed

1 byte, read-only.

This attribute indicates whether the associated software image is committed (1) or uncommitted (0).
This is derived from the U-Boot environment variable `commit_bank`.

Is Active

1 byte, read-only.

This attribute indicates whether the associated software image is active (1) or inactive (0).
This derived from the U-Boot environment variable `active_bank`.

Is Valid

1 byte, read-only.

This attribute indicates whether the associated software image is valid (1) or invalid (0).
This is derived from the U-Boot environment variable `img_validA` for managed entity ID 0 and from `img_validB` for the managed entity ID 1.

Product Code

25 bytes, read-only.

This attribute provides a way for a vendor to indicate product code information on a file. It is a character string, padded with trailing nulls when it is shorter than 25 bytes.

Note: This optional attribute is not part of the reference implementation and can be added by OEM software.

Image Hash

16 bytes, read-only.

This attribute is an MD5 hash of the software image. It is computed at completion of the end download action.

Note: This optional attribute is not part of the reference implementation and can be added by OEM software.

Actions

- Get
  The values of the attributes are reported to the OLT.
- Start download
  The software download process is started.
- Download section
  One section of the software image is downloaded.
- End download
  The software download process is complete. No more section is sent.
- Activate image
  This action selects one of the two images by creating a temporary U-Boot environment variable `img_activate` with an ASCII value of “A” or “B”. After this a reboot is triggered that boots the selected image. During the boot process, the temporary variable is automatically deleted. Permanent variables are not modified.
- Commit image
  This action changes the value of the U-Boot environment variable commit_bank to either “A” or “B”. This is evaluated during the next and subsequent boot processes.

Attribute Value Change Notifications

These attribute changes are reported (see **Table 34**):

- AVC #1: Version
- AVC #2: Is committed
- AVC #3: Is active
- AVC #4: Is valid
- AVC #5: Product code
  This is not implemented by the reference code and must be added by the system vendor, if needed.
- AVC #6: Image hash
  This is not implemented by the reference code and must be added by the system vendor, if needed.

Message Type Selection

When the ONU supports both baseline messages and extended messages types, the OLT selects the type to be used for the software download.

When the OLT starts using baseline messages, the ONU must also respond with baseline messages.

When the OLT starts using extended messages, the ONU must also respond with extended messages.

The OLT is expected to stick to the initially selected message type and not change the type during the software download process.

4.20.1.10 Cardholder

ME class: 5

This managed entity is autonomously created by the ONU and does not require hardware programming.

The reference implementation assumes an integrated ONU device and does *not* support pluggable units, such as optical modules.

Managed Entity ID

2 bytes, read-only.

The ONU sets the first byte of this two-byte identifier to 0 when the ONU contains pluggable equipment modules. For an integrated ONU, the first byte is set to 1.

The second byte of this identifier is the slot number. For an integrated ONU, this byte is either used as a virtual slot or set to 0.

Actual Plug-in Unit Type

1 byte, read-only.

This attribute is equal to the type of the circuit pack in the cardholder, or 0 when the cardholder is empty. When the cardholder is populated, this attribute is the same as the type attribute of the corresponding **Circuit Pack** managed entity.

The value is defined in the MIB initialization file.

The target applications support these types of interfaces:

- 24 - 10/100BASE-T Ethernet
- 32 - POTS
- 45 - Mixed interfaces
- 47 - 10/100/1000BASE-T Ethernet
- 48 - Virtual Ethernet interface point
- 49 - 10G Ethernet
- 50 - 2.5GBASE-T Ethernet
- 234 - TWDM10G2
- 235 - TWDM10G10
- 237 - XG-PON10G2488
- 238 - XG-PON10G10
- 248 - G-PON2488G1244

Note: The codepoint definitions may be updated by ITU to cover new applications.
*For reference see **[56]**. Table 9.1.5-1.*

Attention: Some OLTs do not accept newer code points (such as 10GBASE-T) although they support these interface types.

Expected Plug-in Type

1 byte, read, write.

This attribute provisions the type of circuit pack for the slot. The value 0 means the cardholder is not provisioned to contain a circuit pack. The value 255 means the cardholder is configured for plug-and-play. Upon instantiation, it is set to 0 by the ONU. For integrated interfaces, this attribute may be used to represent the type of interface.

The value is defined in the MIB initialization file.

Expected Port Count

1 byte, read, write.

This attribute allows the OLT to specify the number of ports it expects in a circuit pack. Prior to provisioning by the OLT, the ONU initializes this attribute to 0.

The value is defined in the MIB initialization file.

Expected Equipment ID

20 bytes, read, write.

This attribute provisions the specific type of expected circuit pack. This attribute applies only to ONUs that do not have integrated interfaces. In some environments, this may contain the expected equipment CLEI code. Upon ME instantiation, the ONU sets this attribute to all spaces.

The value is defined in the MIB initialization file.

Actual Equipment ID

20 bytes, read-only.

This attribute identifies the specific type of circuit pack, when it is installed. This attribute applies only to ONUs that do not have integrated interfaces. In some environments, this may include the equipment CLEI code. When the slot is empty or the equipment ID is not known, this attribute must be set to all spaces.

The value is defined in the MIB initialization file, as pluggable units are not supported.

Protection Profile Pointer

1 byte, read-only.

This attribute specifies an equipment protection profile that may be associated with the cardholder. Its value is the least significant byte of the managed entity ID of the equipment protection profile with which it is associated, or 0 when equipment protection is not used.

This value must be set to 0 in the MIB initialization file.

Invoke Protection Switch

1 byte, read, write.

The OLT uses this attribute to control equipment protection switching.

Protection switching is not supported.

Actions

Get, set

Attribute Value Change Notifications

- #1: Actual plug-in type has changed
  In an integrated ONU, this value cannot change.
- #5: Actual equipment ID
  In an integrated ONU, this value cannot change.
- #8: ARC timer has expired

Alarm Notifications

These alarms are defined for this managed entity:

- #0: Plug-in circuit missing
- #1: Plug-in type mismatch
- #2: Improper card removal
- #3: Plug-in equipment ID mismatch
- #4: Protection switch

These alarms are not supported, because they are not applicable to an integrated ONU.

4.20.1.11 Circuit Pack

ME class: 6

This managed entity is autonomously created by the ONU and does not require hardware programming.

Managed Entity ID

2 bytes, read-only.

This value is the same as that of the **Cardholder** managed entity containing this circuit pack instance.

Plug-in Unit Type

The value is defined in the MIB initialization file.

Based on the 10G PON Chipset, target applications are built to support these types of interface:

- 24 - 10/100BASE-T Ethernet
- 32 - POTS (plain old telephone service)
- 39 - LCT (local craft terminal)
- 47 - 10/100/1000/2500/5000/10000/25000/40000BASE-T Ethernet
- 48 - VEIP (virtual Ethernet interface point)
- 49 - 10G Ethernet
- 50 - 2.5GBASE-T Ethernet
- 234 - TWDM10G2
- 235 - TWDM10G10
- 237 - XG-PON10G2488
- 238 - XG-PON10G10
- 248 - G-PON optical interface
- 255 - Plug-and-Play/Unknown

The same code points defined in the **Cardholder** managed entity apply here.

Note: The codepoint definitions may be updated by ITU to cover new applications.

Attention: Some OLTs do not accept newer code points (such as 10GBASE-T) although they support these interface types.

Number of Ports

1 byte, read-only.

This attribute is the number of access ports on the circuit pack. When the port mapping package is supported for this circuit pack, this attribute must be set to the total number of ports of all types.

The value is defined in the MIB initialization file.

Serial Number

8 bytes, read-only.

The serial number is expected to be unique for each circuit pack, at least within the scope of the given vendor.

The serial number may contain the vendor ID and/or version number. For integrated ONUs, this value is identical to the value of the **ONU-G** managed entity serial number attribute. Upon creation, in the absence of a physical circuit pack, this attribute comprises all spaces.

The value is defined in the MIB initialization file.

Version

14 bytes, read-only.

This attribute is a string that identifies the circuit pack version as defined by the vendor. The value 0 indicates the version information is not available or applicable. For integrated ONUs, this value is identical to the value of the **ONU-G** managed entity version attribute. Upon creation, in the absence of a physical circuit pack, this attribute comprises all spaces.

The value is defined in the MIB initialization file.

Vendor ID

4 bytes, read-only.

This attribute identifies the vendor of the circuit pack. For ONUs with integrated interfaces, this value is identical to the value of the **ONU-G** managed entity vendor ID attribute. Upon creation, in the absence of a physical circuit pack, this attribute comprises all spaces.
The value is defined in the MIB initialization file.

Administrative State

1 byte, read, write.

This attribute locks (1) and unlocks (0) the functions performed by this managed entity.

Operational State

1 byte, read-only.

This attribute indicates whether the circuit pack is capable of performing its function. Valid values are enabled (0), disabled (1) and unknown (2). Pending completion of initialization and self-test on an installed circuit pack, the ONU sets this attribute to 2.

Bridged or IP Indication

1 byte, read, write.

This attribute specifies whether an Ethernet interface is bridged or derived from an IP router function. This attribute is applicable only to circuit packs with Ethernet interfaces.

- 0: Bridged
- 1: IP routed
- 2: Bridged and IP routed

The value is defined in the MIB initialization file.

Equipment ID

20 bytes, read-only.

This attribute is used to identify the vendor specific type of circuit pack. In some environments, this attribute may include the CLEI code. Upon instantiation, the ONU sets this attribute to all spaces or to the equipment ID of the circuit pack physically present.
The value is defined in the MIB initialization file.

Card Configuration

1 byte, read, write, set-by-create (when applicable).

This is not supported by the reference software.

Total T-CONT Buffer Number

1 byte, read-only.
Call the `fapi_pon_cap_get` function (`alloc_ids`) to check the number of supported T-CONT buffers. This also includes one dedicated T-CONT for the OMCC.

Note: Sharing of the OMCC GEM/XGEM port with user data as defined for ITU-T G.984.3 is not supported.

Total Priority Queue Number

1 byte, read-only.
Call the `fapi_pon_cap_get()` function to check the number of supported priority queues. This also includes one dedicated queue for the OMCC.

Total Traffic Scheduler Number

1 byte, read-only.

Scan the MIB initialization file to check the number of supported traffic schedulers. This value must be incremented by one scheduler per UNI port not managed by the OMCI, but internally required to handle the downstream priority queues, and another scheduler to handle the OMCC queue.

Power Shedding Override

4 bytes, read, write.

This attribute allows ports to be excluded from the power shed control defined in **Section 4.20.1.12** (**ONU Power Shedding**). It is a bit mask that takes port 1 as the MSB. A bit value of 1 marks the corresponding port to override the power shed timer.

When the circuit pack **Plug-in Unit Type** value is related to data interfaces (code points 24 or 48), there are up to four LAN ports subject to power shedding.

When the circuit pack **Plug-in Unit Type** value is related to voice interfaces (code point 32), there are up to two voice ports subject to power shedding.

For other types, this attribute is not applicable.

Actions

Get, set, create, delete, reboot, test.

Attribute Value Change Notifications

- AVC#7: Operational state change

Alarm Notifications

These alarms are defined for this managed entity:

- #0: Equipment alarm (vendor-specific implementation is required)
- #1: Powering alarm (vendor-specific implementation is required)
- #2: Self test failure (vendor-specific implementation is required)
- #3: Laser end-of-life (vendor-specific implementation is required)
- #4: Temperature yellow
- #5: Temperature red

For an integrated ONU, these alarms are implemented in the same way as the **ONU-G Alarm Notifications**.

4.20.1.12 ONU Power Shedding

ME class: 133

This managed entity is autonomously created by the ONU.

Power shedding allows individual local functions to be disabled, to save power and extend the operational time in case of battery-based operation. These shedding classes are supported by the device:

- Data
  PPTP Ethernet UNI, up to four individual ports
  – Coding: 24 (10/100BASE-T) or 47 (10/100/1000BASE-T)

For each port, power shedding is either enabled or disabled by the OLT. When enabled, the related interface class is automatically switched off upon power failure after the time configured through the related shedding interval attribute and is enabled again when the mains power is recovered.

As the detection of a power failure is application dependent, the software implementation must be adapted accordingly. The OMCI power shedding software must register a callback informing about power loss and power recovery. Based on this callback, the power shedding timer is started or stopped and the affected interfaces are switched off an on, respectively.

Attention: This managed entity is not applicable to applications that do not provide a battery backup function. The reference implementation only provides a reference code and does not support battery backup applications.

Upon managed entity instantiation, all attributes are set to 0.

Managed Entity ID

2 bytes, read-only.

The value is fixed to 0.

Restore Power Timer Reset Interval

2 bytes, read, write

This is the time delay, in seconds, before the power-shedding timers are reset after full power restoration. The start of the interval is determined by the detection of a power recovery event causing the deactivation of backup battery. The handling is application dependent. The battery status information interface is provided through GPIO signals.

Data Class Shedding Interval

2 bytes, read, write.

This time value, given in multiples of one second, defines how long the data interfaces must remain functional after a loss of mains power. When the power loss time exceeds this value, the power saving state is entered and interfaces are disabled.

To enable power saving on the Ethernet interfaces, the Ethernet PHY must be switched off after the configured power shedding interval has elapsed. See the **Power Shedding Override** attribute of the **Circuit Pack** managed entity to find the related LAN ports that must not be subject to power shedding.

Voice Class Shedding Interval

2 bytes, read, write.

This time value, given in multiples of one second, defines how long the integrated voice interfaces must remain functional after a loss of mains power. When the power loss time exceeds this value, the power saving state is entered and interfaces are disabled.

To enable power saving on the voice interfaces, the line feed current is switched off by using `IFX_TAPI_LINE_FEED_SET` with a parameter of `IFX_TAPI_LINE_FEED_HIGH_IMPEDANCE`.

Video Overlay Class Shedding Interval

2 bytes, read, write.

This attribute is not supported. The value is set to 0 by default, any change by the OLT is accepted but ignored. Vendor-specific enhancements are required for the ONU application which offers video overlay functionality.

Video Return Class Shedding Interval

2 bytes, read, write.

This attribute is not supported. The value is set to 0 by default, any change by the OLT is accepted but ignored. Vendor-specific enhancements are required for the ONU application which offers video overlay functionality.

DSL Class Shedding Interval

2 bytes, read, write.

This attribute is not supported. The value is set to 0 by default, any change by the OLT is accepted but ignored. Vendor-specific enhancements are required for the ONU application which offers DSL functionality.

ATM Class Shedding Interval

2 bytes, read, write.

This attribute is not supported. The value is set to 0 by default, any change by the OLT is accepted but ignored.

CES Class Shedding Interval

2 bytes, read, write.

This attribute is not supported. The value is set to 0 by default, any change by the OLT is accepted but ignored.

Frame Class Shedding Interval

2 bytes, read, write.

This attribute is not supported. The value is set to 0 by default, any change by the OLT is accepted but ignored.

SDH/SONET Class Shedding Interval

2 bytes, read, write.

This attribute is not supported. The value is set to 0 by default, any change by the OLT is accepted but ignored.

Shedding Status

This attribute reports the power shedding function status. It is a 16-bit value with only the two most significant bits being used. When a bit is set (1), the power shedding function is active.

- Bit 15: Data class power shedding status
- Bit 14: Voice class power shedding status
- Bit 13 to 0: Set to 0.

Actions

Get, set

Attribute Value Change Notifications

- AVC#11: Shedding status (vendor-specific implementation is required)

Alarm Notifications

There are no alarms defined for this managed entity.

4.20.1.13 Port Mapping Package

ME class: 161

This managed entity is autonomously created by the ONU and does not require hardware programming.

Refer to ITU-T G.988 **[56]** for implementation details.

Attention: A template implementation is available for reference, the functionality must be added by the system vendor.

*Note: The managed entity name is Port Mapping Package in ITU-T G.988 **[56]** and Port Mapping Package-G in ITU-T G.984.4 **[52]**.*

4.20.1.14 Equipment Extension Package

ME class: 160

This managed entity is autonomously created by the ONU and provides functionality implemented by the application through the use of GPIO pins. The implementation is for reference only and requires application-specific enhancement.

A template implementation is available for reference. The functionality must be added by the system vendor.

Managed Entity ID

2 bytes, read-only.

Through an identical ID, this managed entity is implicitly linked to an instance of the **ONU-G** or **Cardholder**.

Environmental Sense

2 bytes, read, write

The hardware programming is application-dependent and uses GPIO inputs for environmental sense input functions. Up to eight inputs are controlled by this managed entity. Use the operating system GPIO driver functions for implementation.

Contact Closure Output

2 bytes, read, write.

The hardware programming is application-dependent and uses GPIO outputs for contact closure output functions. Up to eight outputs are controlled by this managed entity. Use the operating system GPIO driver functions for implementation.

Actions

Get, set

Attribute Value Change Notifications

There are no attribute value changes defined for this managed entity.

Alarm Notifications

These alarms are defined for this managed entity, they are application specific and not part of the reference software implementation:

- #1: Sense point 1 alarm
- …
- #8: Sense point 8 alarm

4.20.1.15 Protection Data

ME class: 279

This managed entity models the capability and parameters of PON protection. An ONU that supports PON protection automatically creates one instance of this managed entity.

One instance of this managed entity is associated with two instances of the **ANI-G**. One of the **ANI-G** managed entities represents the working side; the other represents the protection side.

Attention: This managed entity is not part of the reference implementation and must be added by the system vendor.

Managed Entity ID

2 bytes, read-only

The default ID for the first managed entity of this type is 0. If there is more than one Protection Data managed entity, they are numbered in ascending order.

Working ANI-G Pointer

2 bytes, read, write

This attribute points to the **ANI-G** that represents the *working* side of a protected PON.

Protection ANI-G Pointer

2 bytes, read, write

This attribute points to the **ANI-G** that represents the *protection* side of a protected PON.

Protection Type

1 byte, read, write

This attribute indicates the type of PON protection. Valid values are:

- 0: 1+1 protection
- 1: 1:1 protection without extra traffic
- 2: 1:1 protection with ability to support extra traffic

Revertive Indication

1 byte, read, write, set-by-create (if applicable)

This attribute indicates whether protection is revertive or non-revertive.

- 0: Non-revertive protection, the path is not automatically switching back after the fault condition has been resolved. This is the recommended default value.
- 1: Revertive protection, the path is automatically switched back after the fault condition has been resolved.

Wait to Restore Time

2 bytes, read, write, set-by-create (if applicable)

In revertive protection mode, this attribute specifies the time, in seconds, to wait after a fault clears before switching back to the working path. The default value is 3 seconds.

Wait to Restore Time

2 bytes, read, write, set-by-create (if applicable)

In revertive protection mode, this attribute specifies the time, in seconds, to wait after a fault clears before switching back to the working path. The default value is 3 seconds.

Actions

Get, set

If applicable: Create, delete

Attribute Value Change Notifications

There are no Attribute Value Changes defined for this managed entity.

Alarm Notifications

There are no Alarms defined for this managed entity.

4.20.1.16 Equipment Protection Profile

ME class: 159

This managed entity supports equipment protection. There can be as many as two protection slots protecting as many as eight working slots. Each of the working and protect cardholder managed entities should refer to the equipment protection profile that defines its protection group. Instances of this managed entity are created and deleted by the OLT.

An ONU should deny pre-provisioning that would create impossible protection groupings because of slot or equipment incompatibilities. In the same way, the ONU should deny creation or addition to protection groups that cannot be supported by the current equipped configuration. Even so, an inconsistent card type alarm is defined, for example, to cover the case of a plug-and-play circuit pack installed in a protection group cardholder that cannot support it.

An instance of this object points to the working and protect cardholders, which in turn point back to this managed entity.

Attention: This managed entity is not part of the reference implementation and must be added by the system vendor.

Managed Entity ID

2 bytes, read-only

This attribute uniquely identifies each instance of this managed entity. The first byte is 0. The second byte is assigned by the OLT, and must be unique and non-zero.

Protect Slot 1/2

1 byte per attribute, read, write, set-by-create

This pair of attributes describes the protecting cardholder entities in an equipment protection group. There can be one or two protecting entities.

- 0: Undefined entry (default), a place-holder if there are fewer than two protecting entities in the protection group.
- 1 to 254: Slot number of the protecting circuit pack.

Working Slot 1-8

1 byte per attribute, read, write, set-by-create

This group of attributes describes the working cardholder entities in an equipment protection group. There can be up to eight working entities.

Working slot 1 is mandatory, other working slots are optional.

- 0: Undefined entry (default), a place-holder if there are fewer than eight working entities in the protection group.
- 1 to 254: Slot number of the working circuit pack.

Protect Status 1/2

1 byte per attribute, read-only

This pair of attributes indicates whether each protection cardholder is currently protecting another cardholder, and if so, which one.

- 0: Not protecting any other cardholder.
- 1 to 254: Slot number of the working cardholder currently being protected by this ME.

Revertive Indication

1 byte, read, write, set-by-create

This attribute specifies whether equipment protection is revertive. The default value 0 indicates revertive switching; any other value indicates non-revertive switching.

Wait to Restore Time

1 byte, read, write, set-by-create

This attribute specifies the time, in minutes, during which a working equipment must be free of error before a revertive switch occurs. It defaults to 0.

Actions

Create, delete, get, set

Attribute Value Change Notifications

- AVC #0: Inconsistent card type

Alarm Notifications

There are no Alarms defined for this managed entity.

4.20.1.17 ONU Remote Debug

ME class: 158

This managed entity is autonomously created by the ONU and enables a communication path between the OLT and the ONU. It allows, for example, to extend the ONU command line interface (CLI) to the OLT. For this implementation, the CLI commands are entered at the OLT as if they were locally entered at the ONU through a physical interface and responses are sent back to the OLT.

When the OLT activates this function, the local command line interface (when one exists) must be disabled to avoid confusion.

Attention: Implementation details are vendor-specific and must be added by the system vendor.

An example implementation is available in …/gpon_omci_onu/src/me/omci_onu_remote_debug.c.

Managed Entity ID

2 bytes, read-only.

The value is fixed to 0.

Command Format

1 byte, read-only.

This attribute selects whether the command is given in ASCII format (0) or binary (1).

The example implementation supports ASCII format only.

Command

25 bytes, write-only.

This is a 25-byte ASCII string. When the command to be given is shorter than 25 bytes, it is null-terminated. When the command is longer than 25 bytes, multiple commands are sent in a sequence. The first ASCII character defines, whether the command is the begin, end, or intermediate section of the target string to be sent:

- 00_H: This is an intermediate part of a command, containing neither start nor end of it.
- 01_H: This is the start of a command which must be continued.
- 02_H: This is the end of a command that completes a prior command section.
- 03_H: This is a single complete command, containing start and end.
- Others: Undefined, do not use.

Reply Table

N bytes, read-only.

This attribute carries the ONU reply to the debug command, in ASCII format.

The get, get next action sequence must be used with this attribute, since its size is unspecified.

Actions

Get, get next, set.

Attribute Value Change Notifications

There are no attribute value changes defined for this managed entity.

Alarm Notifications

There are no alarms defined for this managed entity.

4.20.1.18 ONU Dynamic Power Management Control

ME class: 336

This managed entity is autonomously created by the ONU and handles the ONU power management.

These power saving modes are defined:

Doze Mode

The optical receiver is fully functional and downstream traffic is forwarded to the UNI. As long as there is no demand for upstream packet transmission, the optical transmitter is inactive and do not send idle GEM/XGEM frames during time slot allocations.

Note: This power saving mode is not supported for XGS-PON and NG-PON2 operation modes.

Cyclic Sleep Mode

While cyclic sleep mode, the optical receiver and transmitter are disabled and no communication between OLT and ONU takes place unless the OLT or local upstream traffic demands a wake-up.

Note: This power saving mode is not supported for XGS-PON and NG-PON2 operation modes.

Watchful Sleep Mode

This is similar to the cyclic sleep mode but the receiver is taken into operation from time to time to check for wake-up messages from the OLT.

Managed Entity ID

2 bytes, read-only.

The value is fixed to 0.

Power Reduction Management Capability

1 byte, read-only.

This attribute provides information about the supported power saving mode(s), implemented as a bit map. Each bit is related to a dedicated mode, and multiple bits can be set. A bit set (1) indicates that the related mode is supported. The power saving mode definitions are:

- Bit 0: Doze mode
- Bit 1: Cyclic sleep mode
- Bit 2: Watchful sleep mode (only with OMCC version 0xA4)
- Bit 3 to 7: Reserved for future use, must be cleared to 0.

For example, these bit combinations are meaningful (refer to ITU-T G.988):

- 0: No power saving mode is supported.
- 1: Only doze mode is supported.
- 2: Only cyclic sleep mode is supported.
- 3: Doze mode and cyclic sleep mode are supported.
- 4: Watchful sleep mode is supported (only with OMCC version 0xA4/).
- 5 to 255: Reserved bit combinations, must not be used (these are ignored).

The power management capabilities must read through `fapi_pon_cap_get` (`features`; `PON_FEATURE_WSLP`, `PON_FEATURE_CSLP`, `PON_FEATURE_DOZE`).

Power Reduction Management Mode

1 byte, read, write.

This attribute enables or disables one or more power saving modes. It is provided as a bit map according to the definition given for the **Power Reduction Management Capability** attribute. A bit set (1) enables the related mode while a bit cleared (0) disables the power saving mode. These settings are accepted:

- 0: No power saving mode is activated (default).
- 1: Doze mode is activated.
- 2: Cyclic sleep mode is activated.
- 3: Invalid, as it ambiguously activates two different modes.
- 4: Watchful sleep mode is activated.
- 5: Invalid, as it ambiguously activates two different modes.
- 6: Invalid, as it ambiguously activates two different modes.
- 7: Invalid, as it ambiguously activates three different modes.
- >7: Reserved, do not use. Must result in disabling any power saving mode (acts as in mode 0).

Itransinit

2 bytes, read-only.

This time value indicates the worst case time for the ONU to return to full functionality after leaving the asleep state in cyclic sleep mode or the low-power state in watchful sleep mode.

The attribute value is fixed by design and given in multiples of 125 µs.

Read this value through `fapi_pon_cap_get`.

Itxinit

2 bytes, read-only.

This time value indicates the time needed for the transmitter initialization when leaving the listen state in the doze power saving mode.

The attribute value is fixed by design and given in multiples of 125 µs.

Read this value through `fapi_pon_cap_get`.

Maximum Sleep Interval

4 bytes, read, write.

This attribute (*Ilowpower*) defines the timer interval to be used by the *Tlowpower* timer, given in multiples of 125 µs.

Configure this value through `fapi_pon_psm_cfg_set` (`max_sleep_interval`).

The value must be greater than or equal to (4 + Twakeup).

Maximum Receiver-off Interval

4 bytes, read, write.

This attribute specifies the maximum time the OLT waits from the moment it decides to wake up an ONU in the low power state of the watchful sleep mode until the ONU is fully operational. The value is given in multiples of 125 µs.

Attention: This attribute has been introduced in ITU-T G.988 Amendmend 1. It breaks the existing earlier implementations, thus it must be implemented depending on the OMCC version (only with 0xA4). For older OMCC versions, this attribute must be omitted.

Minimum Aware Interval

4 bytes, read, write.

This attribute (*Iaware*) defines the minimum time that must be spent in an aware state of the power saving state machine. This value is used to initialize the *Taware* timer, given in multiples of 125 µs.

Configure this value through `fapi_pon_psm_cfg_set` (`min_aware_interval`).

This value must be smaller than (2^23-1 - Twakeup).

Minimum Active Held Interval

2 bytes, read, write.

The attribute (*Ihold*) specifies the *Thold* timer load value. It defines the minimum time during which the ONU remains in the active held state and is given in multiples of 125 µs.

Configure this value through `fapi_pon_psm_cfg_set` (`min_active_held_interval`)

Maximum Sleep Interval Extension

8 bytes, read, write.

This attribute designates maximum sleep interval values for doze mode and cyclic sleep mode separately.

When it supports this attribute, the ONU ignores the value of the maximum sleep interval attribute.

Maximum sleep interval for doze mode: 4 byte

This defines the maximum time the ONU spends in its listen state, given in multiples of 125 µs.

Maximum sleep interval for cyclic sleep mode: 4 byte

This defines the maximum time the ONU spends in its asleep state, given in multiples of 125 µs.

Configure this value through `fapi_pon_psm_cfg_set` (`max_sleep_interval_doze_ext`, `max_sleep_interval_cyclic_sleep_ext`)

EPON Capability Extension

1 byte, read-only.

This attribute declares the EPON specific capabilities for the dynamic power management control.

It must report a value of 0, the EPON-specific power saving modes are not supported.

This attribute is not present for OMCC versions below 0xA4 and not required for the ITU PON applications.

EPON Setup Extension

1 byte, read, write.

This attribute specifies the EPON specific configurations for the dynamic power management control.

This attribute is ignored.

This attribute is not present for OMCC versions below 0xA4 and not required for the ITU PON applications.

Missing Consecutive Bursts Threshold

4 bytes, read, write.

The $Clob_i$ attribute specifies the maximum number of missing consecutive scheduled bursts from the ONU that the OLT tolerates without raising an alarm. The value of this attribute defaults to 4 (500 ms).

There is no hardware configuration required for this value.

This attribute is not present for OMCC versions below 0xA4 and not required for the ITU PON applications.

Actions

Get, set

Attribute Value Change Notifications

There are no attribute value changes defined for this managed entity.

Alarm Notifications

There are no alarms defined for this managed entity.

4.20.1.18.1 Power Saving State Machine Implementation

The state machine to handle power saving modes is implemented by the PON IP firmware. Communication between the OMCI software and the firmware is provided by the PON library functions.

Timers

There are three timers dedicated to the power saving handling and used to control the state transitions of the power saving state machine when they expire.

The timer configuration is configured through the `fapi_pon_psm_cfg_set` function.

- Taware
  - This timer is restarted each time the state machine enters one of the aware states (sleep aware or doze/watch aware. It expires after Iaware * 125 µs.
- Thold
  - This timer is restarted each time a sleep request/awake PLOAM message is received to send the state into the active held state. It expires after Ihold * 125 µs.
- Tlowpower
  - This timer is restarted each time the state machine enters one of the low power states (asleep, listen, or watch)

Counters

These are the counters provided to support the supervision of the energy saving states:

- Doze time counter
  Counts the 125 µs intervals that are spent in doze mode.
- Cyclic-sleep time counter
  Counts the 125 µs intervals that are spent in cyclic-sleep mode.
- Watchful-sleep time counter
  Counts the 125 µs intervals that are spent in watchful-sleep mode.

The counters are readable through the `fapi_pon_psm_counters_get` function.

4.20.1.19 Energy Consumption Performance Monitoring History Data

ME class: 343

This managed entity is autonomously created by the ONU and provides a summary of the time spent in one of the power saving modes and the accumulated power that has been used. Energy consumption reporting needs external circuitry to measure the current drawn on the main supply and the main supply voltage. This is vendor-specific and not provided as part of the software delivery.

Managed Entity ID

2 bytes, read-only

The value is fixed to 0.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

No thresholds are defined for this managed entity, a null pointer must be assigned to this attribute.

The attribute itself can be ignored, whatever pointer is assigned.

Doze Time

4 bytes, read-only

This attribute returns the time that the ONU has spent in “doze mode”, if this has been activated by setting the **Power Reduction Management Mode** of the **ONU Dynamic Power Management Control** managed entity.

Use `fapi_pon_psm_counters_get` (`doze_time`) to retrieve the value.

The value is given in units of 1 µs.

Cyclic Sleep Time

4 bytes, read-only

This attribute returns the time that the ONU has spent in “cyclic sleep mode”, if this has been activated by setting the **Power Reduction Management Mode** of the **ONU Dynamic Power Management Control** managed entity.

Use `fapi_pon_psm_counters_get` (`cyclic_sleep_time`) to retrieve the value.

The value is given in units of 1 µs.

Watchful Sleep Time

4 bytes, read-only

This attribute returns the time that the ONU has spent in “watchful sleep mode”, if this has been activated by setting the **Power Reduction Management Mode** of the **ONU Dynamic Power Management Control** managed entity.

Use `fapi_pon_psm_counters_get` (`watchful_sleep_time`) to retrieve the value.

The value is given in units of 1 µs.

Energy Consumed

4 bytes, read-only

No implementation is included in the software delivery, a constant value of 0 mJ is returned.

This attribute is subject to vendor-specific enhancements and requires additional hardware to measure the supply voltage and supply current.

Actions

Create, delete, get, set, get current data

Attribute Value Change Notifications

There are no Attribute Value Changes defined for this managed entity.

Alarm Notifications

There are no alarms defined for this managed entity.

4.20.2 ANI and Traffic Management

These managed entities are defined by ITU-T G.988 **[56]** in chapter 9.2.

4.20.2.1 ANI-G

ME class: 263

This managed entity is autonomously created by the ONU and configures the network-side G-PON interface (access network interface). The first byte is used as slot ID, the second byte is used as port ID.

Managed Entity ID

2 bytes, read-only.

The value is defined in the MIB initialization file. The first byte is used as slot ID, the second byte is used as port ID.

SR Indication

1 byte, read-only.

Status reporting is supported by hardware. When the application offers status reporting, this value must be set to true (1), otherwise false (0).

Use `fapi_pon_cap_get` to read the `features` bit map. When one of the bits `PON_FEATURE_DBAM0` or `PON_FEATURE_DBAM1` is set, reports `true`; otherwise reports `false`.

The MIB initialization data is ignored.

Total T-CONT Number

2 bytes, read-only.

The number of T-CONTs present in the OMCI MIB is counted and compared with the value retrieved by using `fapi_pon_cap_get` (`alloc_ids`). The smaller number is reported.

Both the **ANI-G** and **T-CONT** managed entities are autonomously created during OMCI initialization. The T-CONT managed entities must be created first for correct reporting of this number.

The value stored in the MIB initialization data is ignored.

GEM Bock Length

2 bytes, read, write.

The GEM block length is a scaling factor required by the PON IP hardware module to report the queue backlog in the correct unit. The configuration is done by calling the PON library function `fapi_pon_gpon_cfg_set`.

The typical values to be used are 4 byte in G.987, G.9807, and G.989 or 48 byte in G.984.

This attribute is ignored, when the **ONU2-G** managed entity provides the newer **Priority Queue Scale Factor** attribute (see **Figure 63**).

This value is read from the MIB initialization data and configured through the `pon_pqsf_cfg_set` function in case of G.984 operation mode only. Other G-PON modes use a fixed hardware reporting scale factor of 4 byte.

Piggyback DBA Reporting

1 byte, read-only.

Use `fapi_pon_cap_get` to read the features bit map. The reporting must be as follow:

- 0: `if ((PON_FEATURE_DBAM0 == 1) && (PON_FEATURE_DBAM1 == 0))`
- 1: `if ((PON_FEATURE_DBAM0 == 1) && (PON_FEATURE_DBAM1 == 1))`
- 4: `if ((PON_FEATURE_DBAM0 == 0)`

The MIB initialization data is ignored.

Deprecated Attribute

1 byte, read-only.

This attribute is no longer used and reports a fixed value of 0, regardless of the MIB initialization data.

SF Threshold

1 byte, read, write.

The signal fail threshold is configured by using `fapi_pon_gtc_cfg_set` (`sf_thr`) and reads back by using `fapi_pon_gtc_cfg_get` (`sf_threshold`).

This default value used for managed entity creation is read from the MIB initialization data.

SD Threshold

1 byte, read, write.

The signal fail threshold is configured by using `fapi_pon_gtc_cfg_set` (`sf_thr`) and reads back by using `fapi_pon_gtc_cfg_get` (`sf_threshold`).

This default value used for managed entity creation is read from the MIB initialization data.

ARC

1 byte, read, write.

This attribute allows the activation of Alarm Reporting Control (ARC) for this managed entity.

- 0: Disabled, alarms are reported, this is the default value.
- 1: Enabled, alarms are suppressed.

Note: Yes, this is indeed inverse logic, defined by ITU!

ARC Interval

1 byte, read, write.

This attribute defines the interval used with the ARC function for this managed entity. The values 0 to 254 give the duration in minutes for the timer which resets the ARC to disabled state after the last alarm has changed to inactive. The special value 255 means that the timer never expires. The default value is zero.

Optical Signal Level

2 bytes, read-only.

This value reports the currently received optical signal level (receive power) in the downstream direction.

The information is accessible through a PON library function `fapi_pon_optic_status_get` (`rx_power`) and is already reported as a 2s complement integer value referred to 1 mW (which is dBm), with 0.002 dB granularity.

By default, it is set to -32768 (lowest possible optical receive power that can be coded).

Lower Optical Threshold

1 byte, read, write.

This threshold is compared with the received optical power value that is given by the optical signal level attribute. An alarm is issued when the threshold is exceeded (receive power is too small).

Valid values are -127 dBm (coded as 254) to 0 dBm (coded as 0) in 0.5 dBm increments.

The default value is read from the MIB initialization data.

The default value 255 selects the ONU automatic setting. It uses a value of -29 dBm, which is below the receiver useful range.

The automatic setting is defined in `/etc/config/optic, lower_receive_optical_threshold`.

See **Table 7** for more details.

Upper Optical Threshold

1 byte, read, write.

This threshold is compared with the received optical power value given by the optical signal level attribute. An alarm is issued when the threshold is exceeded (receive power is too high).

Valid values are -127 dBm (coded as 254) to 0 dBm (coded as 0) in 0.5 dBm increments.

The default value is read from the MIB initialization data.

The default value 255 selects the ONU automatic setting. It uses a value of -7 dBm, which is above the receiver useful range.

The automatic setting is defined in `/etc/config/optic, upper_receive_optical_threshold`.
See **Table 7** for more details.

ONU Response Time

2 bytes, read-only.

This value is configured in the PON IP hardware module and read through `fapi_pon_gpon_status_get` (`onu_resp_time`). The valid range is from 34000 (34 µs) to 36000 (36 µs), with a typical value of about 35000 (35 µs).

Transmit Optical Level

2 bytes, read-only.

This value reports the currently transmitted optical signal level (transmit power) in the upstream direction.

The information is accessible through a PON library function `fapi_pon_optic_status_get` (`tx_power`) and already reported as a 2s complement integer value referred to 1 mW (which is dBm), with 0.002 dB granularity.

The default value is read from the MIB initialization data.

When the information cannot be read from the optical module through `fapi_pon_optic_status_get`, it is set to -32768 (lowest possible optical transmit power that can be coded).

Lower Transmit Power Threshold

1 byte, read, write.

This threshold is compared with the transmitted optical power value given by the transmit optical level attribute. An alarm is issued when the threshold is exceeded (transmit power is too small).

The threshold value is a 2s complement integer given in dBm, with 0.5 dBm granularity.

The default value of -63.5 dBm (0x81) selects the ONU automatic setting. It must use a value below the ONU minimum useful transmit power, which is -7 dBm (such as +1 dBm).

The automatic setting is defined in `/etc/config/optic, lower_transmit_power_threshold`.
See **Table 7** for more details.
The default value is read from the MIB initialization data.

Upper Transmit Power Threshold

1 byte, read, write.

This threshold is compared with the transmitted optical power value given by the transmit optical level attribute. An alarm is issued when the threshold is exceeded (transmit power is too high).

The default value of -63.5 dBm (0x81) selects the ONU automatic setting. It must use a value above the ONU maximum useful transmit power, which is +4 dBm (such as +10 dBm).

The automatic setting is defined in `/etc/config/optic, upper_transmit_power_threshold`.
See **Table 7** for more details.
The default value is read from the MIB initialization data.

Actions

Get, set, test

Note: The test function reports the optical power values in units of dBµ (referenced to 1 µW) while the values provided by the previous attributes are reported in units of dBm (referenced to 1 mW; 1 dBm = 30 dBµ).

AVC Notifications

AVC #8: ARC cancellation

Alarm Notifications

These alarms are defined for this managed entity:

- 0: Low received optical power
  - The received downstream optical power is below the configured threshold.
- 1: High received optical power
  - The received downstream optical power is above the configured threshold.
- 2: Signal fail (SF)
  - The downstream bit error rate is above the configured SF threshold.
- 3: Signal degrade (SD)
  - The downstream bit error rate is above the configured SD threshold.
- 4: Low transmitted optical power
  - The transmitted upstream optical power is below the configured threshold.
- 5: High transmitted optical power
  - The transmitted upstream optical power is above the configured threshold.
- 6: High laser bias current
  - The laser bias current is above the configured threshold. The configuration is an application-specific fixed value, used to indicate that the laser reaches the end of its usable lifetime.
    The threshold value is defined in `/etc/config/optic` as option `bias_threshold`.
    The unit uses multiples of 2 µA.

Alarm Handling

The optical interface information of receive and transmit power as well as bias current and temperature are read regularly by the OMCI daemon (omcid).

The read frequency is defined in the `pon_lib` file `fapi_pon_me_ani_g_alarm.c` in units of 1 second:

`#define OPTIC_CHECK_INTERVAL 10`The receive and transmit power levels as well as the laser bias current are compared with the related upper and lower thresholds and alarms are sent to the OLT when any of the thresholds is exceeded.

The temperature is handled in a similar way, contained in the **ONU-G** managed entity.

The SD (signal degrade) and SF (signal fail) alarms are sent when the related firmware event message is received through the PON adapter software module.

4.20.2.2 ANI2-G

ME class: 467

This managed entity is autonomously created by the ONU and configures the network-side G-PON interface (access network interface). The first byte is used as slot ID, the second byte is used as port ID.

This managed entity is a supplement of the ANI-G.

Note: This managed entity is not supported by the reference implementation.

Managed Entity ID

2 bytes, read-only.

The value is defined in the MIB initialization file. The first byte is used as slot ID, the second byte is used as port ID. It is the same value as used for the ANI-G managed entity.

Lower Bound Transmit Wavelength

2 bytes, read-only.

This attribute specifies the minimum wavelength (lower bound) that the ONU may transmit over the covered operation temperature range. The value is given in units of nm. If this attribute has the value 0, the lower bound transmit wavelength is undefined.

Upper Bound Transmit Wavelength

2 bytes, read-only.

This attribute specifies the maximum wavelength (upper bound) that the ONU may transmit over the covered operation temperature range. The value is given in units of nm. If this attribute has the value 0, the upper bound transmit wavelength is undefined.

Actions

Get

Notifications

This managed entity does not generate notifications.

4.20.2.3 T-CONT

ME class: 262

The T-CONT (traffic container) managed entities are created by the ONU upon OMCI initialization. The number of T-CONTs is limited by the number of instances provided in the MIB initialization data. This value must not excess the physical capability.

The managed entity ID of each T-CONT is not identical to the T-CONT hardware index, internally used by the PON IP hardware module, and the QoS index, used by the QoS hardware module. A reference table held in the OMCI context maps the managed entity IDs to the T-CONT index and QoS index values.

Starting from predefined base values, the T-CONT and QoS index values are assigned during T-CONT managed entity creation in ascending order.

The base T-CONT index is 1. Index 0 is used for the OMCI channel and thus reserved.

The base QoS index is retrieved from the PON data path driver. This index base corresponds to the dequeue port base in the data path driver context.

Before creating another T-CONT managed entity, the number of available hardware instances must be checked (`fapi_pon_cap_get`, `alloc_ids`).

To create a new T-CONT managed entity, call `fapi_pon_alloc_id_add` with the given `alloc_id` value. The function returns the related `allocation_index`, as assigned by the firmware.

To delete an existing T-CONT managed entity, call `fapi_pon_alloc_id_delete` with the given `alloc_id` value.

Managed Entity ID

2 bytes, read-only.

This number indicates the physical capability that realizes the T-CONT. It may be represented as `0xSSBB`, where SS indicates the slot ID containing this T-CONT (0 for the ONU as a whole), and BB is the T-CONT ID, numbered by the ONU itself. The T-CONTs are numbered in ascending order, with a range from 0 to 255 in each slot.

Alloc-ID

2 bytes, read, write

The allocation IDs are initially assigned to an ONU through the PLOAM channel. The software selects the next free T-CONT managed entity ID, when the Alloc-ID has never been used before. The resulting configuration is read back through the T-CONT OMCI managed entity (one managed entity per T-CONT index).

To assign an allocation ID to a T-CONT, the PON IP hardware must be configured through the `fapi_pon_alloc_id_add` PON library function.

The T-CONT used for the OMCI upstream transmission is provisioned through PLOAM and not visible to OMCI.

To modify an existing T-CONT managed entity, call `fapi_pon_alloc_id_get` with the given `alloc_id` value to retrieve the related allocation index. Then, call `fapi_pon_alloc_index_set` with the `alloc_index` and `alloc_id` values to update the allocation ID.

The provisioned allocation IDs are removable either through the PLOAM protocol by sending an assign Alloc ID message with Alloc ID type = 255 or through OMCI by assigning an Alloc ID of `0x00FF` or `0xFFFF` to the T-CONT. The hardware configuration is updated by using `fapi_pon_alloc_id_delete` (`alloc_id`).

Upon creation of this managed entity, the default value is read from the MIB initialization data. The value must be set to `0xFFFF`, indicating that the T-CONT is initially unassigned.

Deprecated Attribute

1 byte, read-only.

This value must always be reported as 1.

Policy

1 byte, read, write.

A **T-CONT** is used with or without a **Traffic Scheduler** to connect to a **Priority Queue** (see **Figure 75**). To support direct connection of queues, a dedicated scheduler hardware module must be attached to the T-CONT hardware resource. This is invisible to the OMCI.

The policy configuration (Null (0), strict priority (1), or WRR (2)) is done by calling the related QoS function for downstream and `tc` API for upstream.

The default value is read from the MIB initialization data.

When the OLT attempts to modify this value, the **ONU2-G** attribute **QoS Configuration Flexibility** is checked (bit 4). When the flexibility is provided (bit 4 == 1), the policy configuration is modified as requested. When not (bit 4 == 0), the OMCI message is answered with parameter error result reason code.

The OMCI creates a T-CONT network device using the PON Ethernet driver `rtnetlink` APIs. The `ip` command is used for that purpose but this works also with `libnl`. All necessary extensions are added to the standard tools and libraries. This command creates T-CONT:

```
ip link add link pon0 tcont33 type tcont id 33
```

Each T-CONT network devices is multi queue capable. This is required for the next configuration step of the scheduling policy of the T-CONT. For this purpose, the OMCI stack must support configuration of `tc-mqprio`, `tc-mq` and `tc-prio qdisc` on this network device. The `qdisc` must to receive the scheduling policy (SP/WRR) as a parameter. For this, the `hw` parameter is used. The stack must also support the `tc-drr` schedulers. **Figure 65** shows the configuration when `tc-mqprio` is used.

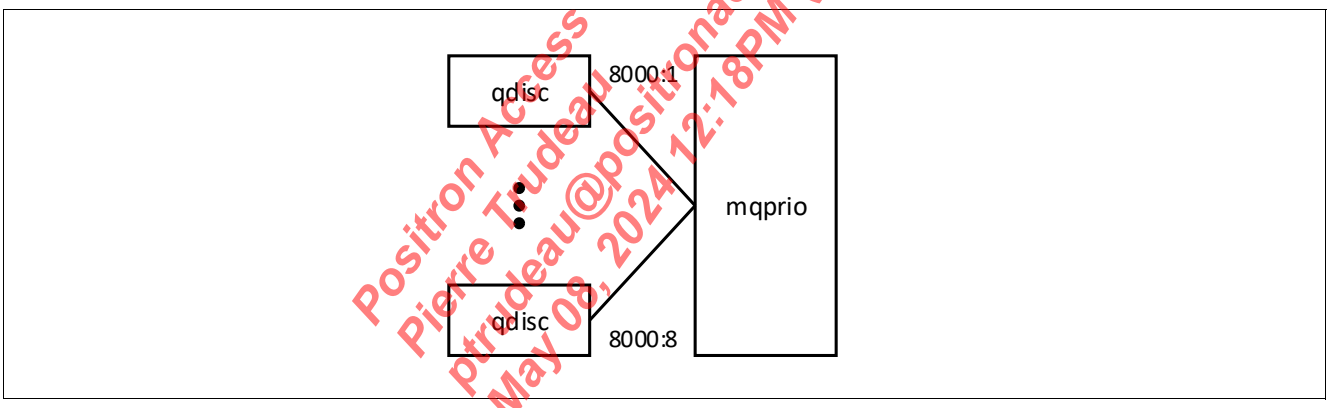


Figure 65 TC-MQPRIO Initial qdisc Tree

The `mqprio` creates up to 16 queues for 16 traffic classes. This information is then used to create the correct scheduling tree and queues. The actual hardware configuration of the queues happens when the OMCI priority queues are created. The queue configuration requires the T-CONT network device with the correct `qdisc` defining the right scheduling policy.

Figure 66 shows the omci stack modules involved in the T-CONT configuration. The figure shows only the update path. The rest of configurations such as the destroy path is very similar. The T-CONT **PON Adapter** operations structures are exported by the `pon_lib` and `pon_net_lib` lower layer modules. The `pon_lib` calls PON library functions to setup the allocation ID in the PON IP firmware. The `pon_net_lib` creates the T-CONT network device and creates the first level traffic scheduler.

Actions

Get, set

Notifications

There are no attribute value changes and no alarms defined for this managed entity.

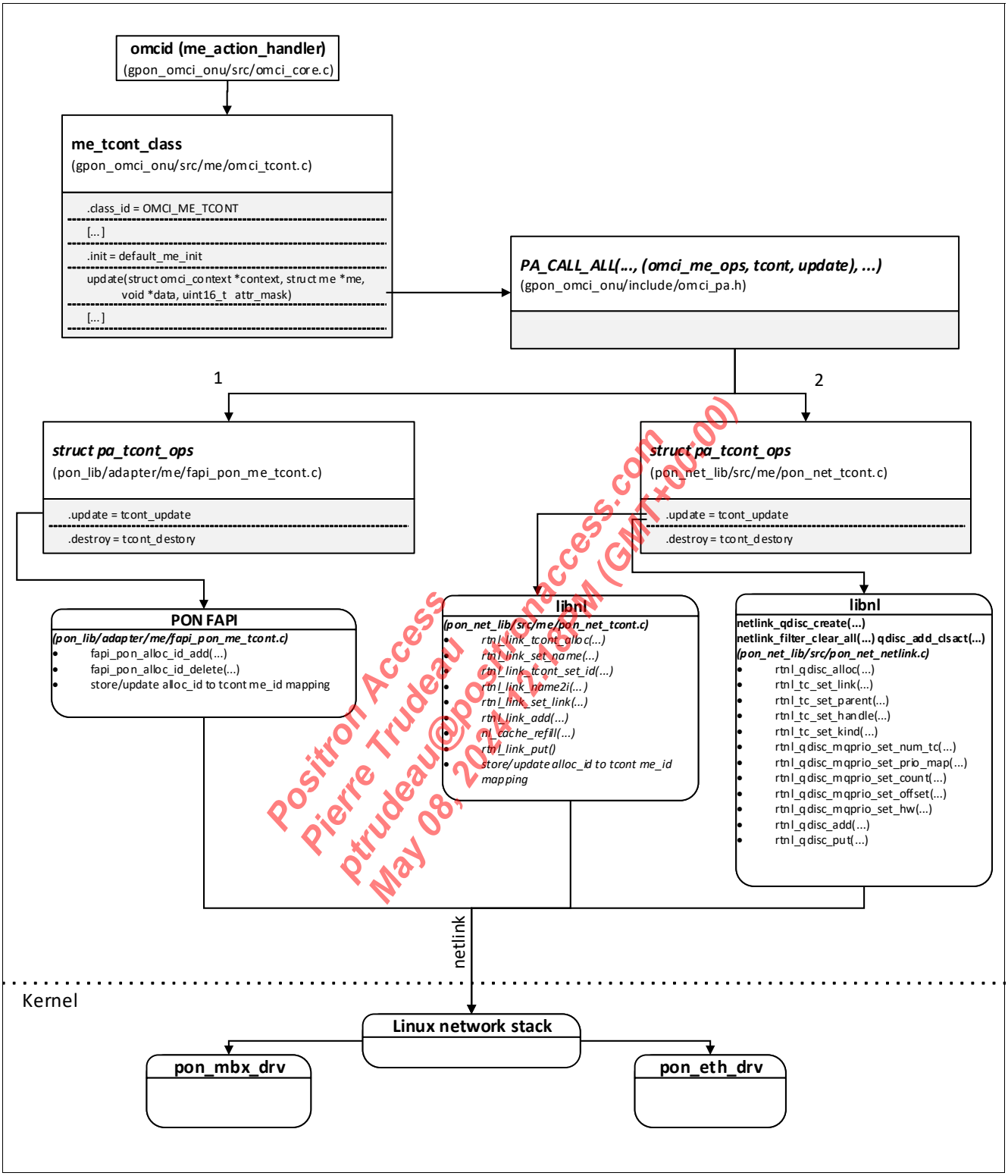


Figure 66 Modules Involved in the OMCI T-CONT Configuration

4.20.2.4 GEM Port Network CTP

ME class: 268

This managed entity is created on the OLT request and assigns a GEM/XGEM port ID to an GEM/XGEM port connection termination point. Furthermore, priority queues and traffic management options are configured.

The port ID value allocates a certain value to this ONU. Within the ONU, multiple Port ID values are handled, each identified by a unique managed entity ID. The **T-CONT** pointer identifies one of the available T-CONTs, each are identified by a unique index. It is possible to assign multiple upstream GEM/XGEM ports to the same T-CONT, but at least one T-CONT must be assigned to allow for upstream data transmission.

Before creating another GEM port network CTP, the number of available hardware instances must be checked (`fapi_pon_cap_get`, `gem_ports`).

See **System Configuration Options** for details.

Managed Entity ID

2 bytes, read-only, set-by-create.

This attribute is a unique number to identify this managed entity, assigned by the OLT.
The OLT uses the same number value as for the GEM/XGEM port ID.

Port ID Value

2 bytes, read, write, set-by-create.

Check whether the GEM port is already defined in the hardware configuration. When it already exists, the existing data is modified, otherwise a new one is configured.

Note: The configuration of GEM/XGEM port ID values is only possible while the connection is in PLOAM activation state O5.x. Otherwise, the configuration attempt is rejected.

T-CONT Pointer

2 bytes, read, write, set-by-create.

The T-CONT allocation for upstream traffic on the GEM/XGEM port is set by using the PON library function `fapi_pon_gem_port_alloc_set` (with `gem_port_id` and `allocation_id`). It is possible to assign multiple GEM/XGEM ports to the same T-CONT.

Direction

1 byte, read, write, set-by-create.

This value controls whether the port ID is registered for upstream, downstream, or both directions.

The unidirectional GEM/XGEM ports in downstream direction are used to carry multicast or broadcast data streams.

The value configured by the OLT is handed over to the PON IP by using the `fapi_pon_gem_port_cfg_set` function (`direction`). This attribute is used in G-PON (ITU-T G.984) systems only and is left unconfigured (0) by the OLT in other PON applications. The related functionality is covered by the **Encryption Key Ring** attribute.

When the port ID is configured for downstream only, no T-CONT and egress queue are assigned to the GEM/XGEM port.

When it is configured for upstream only, the GTC downstream filter table is not configured with this port ID in the PON IP.

Encryption State

1 byte, read-only.

The encryption status is controlled by the PLOAM process and stored in the PON IP hardware. The legal values are defined to be the same as those of the security mode attribute of the **ONU2-G**, with the exception that attribute value 0 indicates an unencrypted GEM/XGEM port.

Traffic Management Pointer for Upstream

2 bytes, read, write, set-by-create.

The handling of this attribute depends on the definition provided through the **Traffic Management Option** attribute of the **ONU-G** managed entity.

ONU is Rate-Controlled

When the traffic management option attribute of the ONU-G managed entity is selected as rate controlled (1), this attribute identifies the T-CONT used for upstream data transmission. **Figure 67** offers a OMCI model for this mode.

The internal implementation requires the use of a queue per GEM CTP and a scheduler per T-CONT to handle the traffic. When up to eight GEM/XGEM ports point to the same T-CONT, a single scheduler is sufficient. When more than eight GEM ports are transmitted in the same T-CONT, hierarchical scheduling must be used. Up to 16 GEM ports are supported by a two-stage scheduling hierarchy.

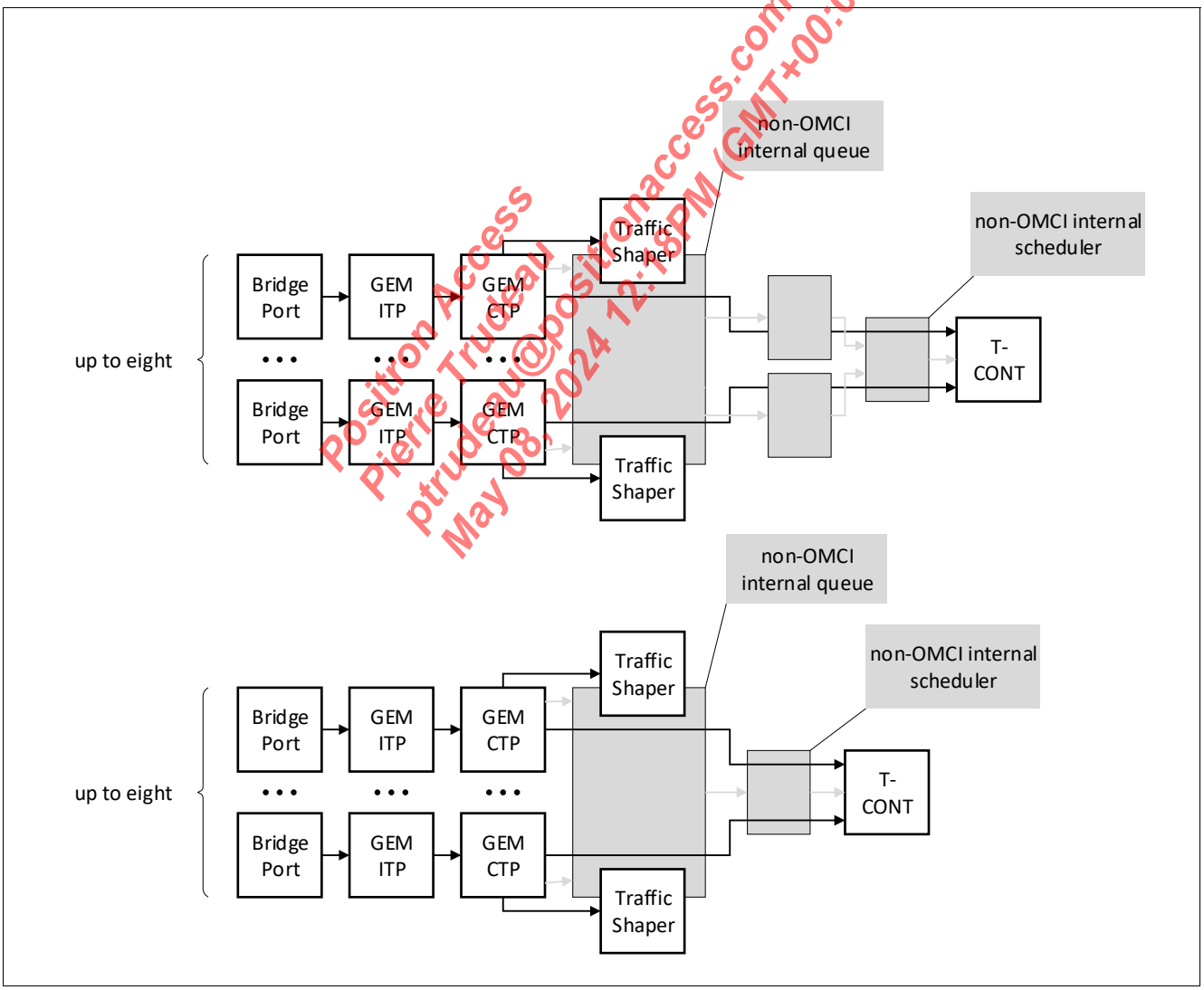


Figure 67 Rate Controlled Traffic Management

ONU is Priority-Controlled or Priority- and Rate-Controlled

When the traffic management option attribute of the ONU-G managed entity is selected as priority controlled (0) or priority and rate controlled (2), this parameter defines the upstream queue-to-GEM-Port relationship within the OMCI model.

Traffic Descriptor Profile Pointer for Upstream

2 bytes, read, write, set-by-create.

ONU is Rate-Controlled

When the traffic management option attribute of the ONU-G managed entity is selected as rate controlled (1), this attribute defines a traffic shaper configuration.
When multiple GEM ports are configured to use a shared queue, this is not supported.

The maximum number of supported GEM ports in Rate-Controlled mode is 32, including the OMCI GEM port.

ONU is Priority-Controlled or Priority- and Rate-Controlled.

When the traffic management option attribute of the ONU-G managed entity is selected as priority and rate controlled (2), this attribute defines a traffic policer configuration.

UNI Counter

1 byte, read-only.

This value is defined by the application and stored in the application specific configuration area of the non-volatile memory.

Priority Queue Pointer for Downstream

2 bytes, read, write, set-by-create.

In case this pointer is not NULL, the priority queue for the GEM downstream traffic must be configured. The scheduler and queue parameters must be configured prior to associating the downstream queue to a GEM port. This is done with the command:

```
tc filter add dev eth0_0 ingress flower skip_sw indev gem1 classid 1:1 action ok
```

The `eth0_0` corresponds to the UNI netdev and the classid 1:1 is the ID of the first queue of the qdisc configured on the UNI netdev.

Traffic Descriptor Profile Pointer for Downstream

2 bytes, read, write, set-by-create.

The upstream queue for the GEM port must be setup. The scheduler and queue parameters must be configured prior to associating the upstream with the GEM port, although reconfiguration in later stage is still possible. This is done with the command:

```
tc filter add dev tcont1 ingress flower skip_sw indev gem1 classid 8000:1 action ok
```

The `tcont1` corresponds to the T-CONT netdev linked to the GEM port and the classid 8000:1 is the ID of the first queue of the qdisc configured on the T-CONT netdev.

Encryption Key Ring

This is an additional attribute which must be implemented for XG-PON, XGS-PON support, and NG-PON2. For G-PON, the OLT does not configure this attribute. In G-PON (ITU-T G.984) systems, only the **Direction** attribute is configured. The functional downstream behavior of the encryption key ring configuration is shown in **Table 44**.

Table 44 Encryption Key Ring Handling Downstream

Configured Encryption Key Ring		Incoming Downstream Key Index	Action
No Encryption	0	0	Pass plain text.
		1	Count a key error, drop the packet.
		2	Count a key error, drop the packet.
		3	Count a key error, drop the packet.
Unicast Encryption Upstream and Downstream	1	0	Pass plain text.
		1	Decrypt, use unicast key #1.
		2	Decrypt, use unicast key #2.
		3	Count a key error, drop the packet.
Multicast Encryption Downstream only	2	0	Pass plain text.
		1	Decrypt, use broadcast key #1.
		2	Decrypt, use broadcast key #2.
		3	Count a key error, drop the packet.
Unicast Encryption Downstream only	3	0	Pass plain text.
		1	Decrypt, use unicast key #1.
		2	Decrypt, use unicast key #2.
		3	Count a key error, drop the packet.

When the value is configured by the OLT, it is handed over to the PON submodule by using the function `fapi_pon_gem_port_cfg_set` (`encryption_key_ring`).

To access this function, perform `fapi_pon_gem_port_cfg_get` to retrieve the current configuration. Then, modify the `encryption_key_ring` parameter of the structure to the value received through the OMCI attribute. This is a one-to-one mapping, identical code points are used by OMCI and the library function.

Figure 68 presents the OMCI daemon modules involved in the data path configuration necessary for the each GEM CTP. The OMCI daemon receives the GEM CTP ME and calls the update callback from the `me_gem_port_network_ctp_class`. This function uses the PA_CALL_ALL macro to call the **PON Adapter** GEM CTP operation hooks in the lower layer modules. The GEM CTP operations are implemented in the `pon_lib` and `pon_net_lib` modules. The first operations in `pon_lib` configure and allocate the GEM CTP in PON IP firmware while the ones in `pon_net_lib` create the GEM port network device and configure the rest of the GEM CTP attributes, for example by assigning a GEM port to an upstream queue.

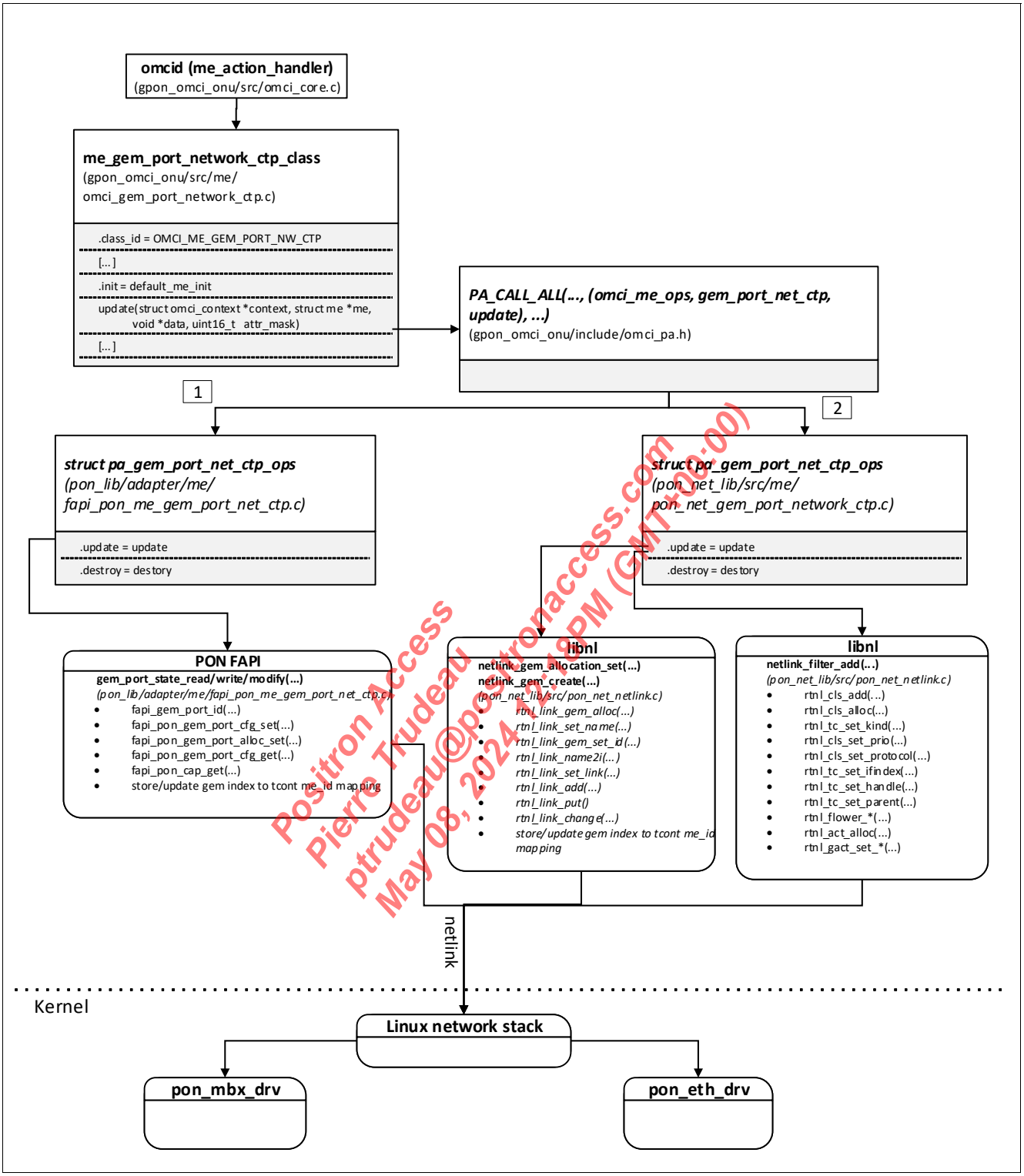


Figure 68 Modules Used in the GEM CTP Configuration

Actions

Create, delete, get, set

Alarm Notifications

These alarms are defined for this managed entity:

- End-to-end loss of continuity

This is an optional alarm, not supported by the software delivery.

4.20.2.4.1 System Configuration Options

This section provides more detail about the system QoS configuration options.

Figure 69 to **Figure 70** show the supported system configuration options.
Figure 71 shows an unsupported option.

Figure 72 depicts the software API flow and dependencies.

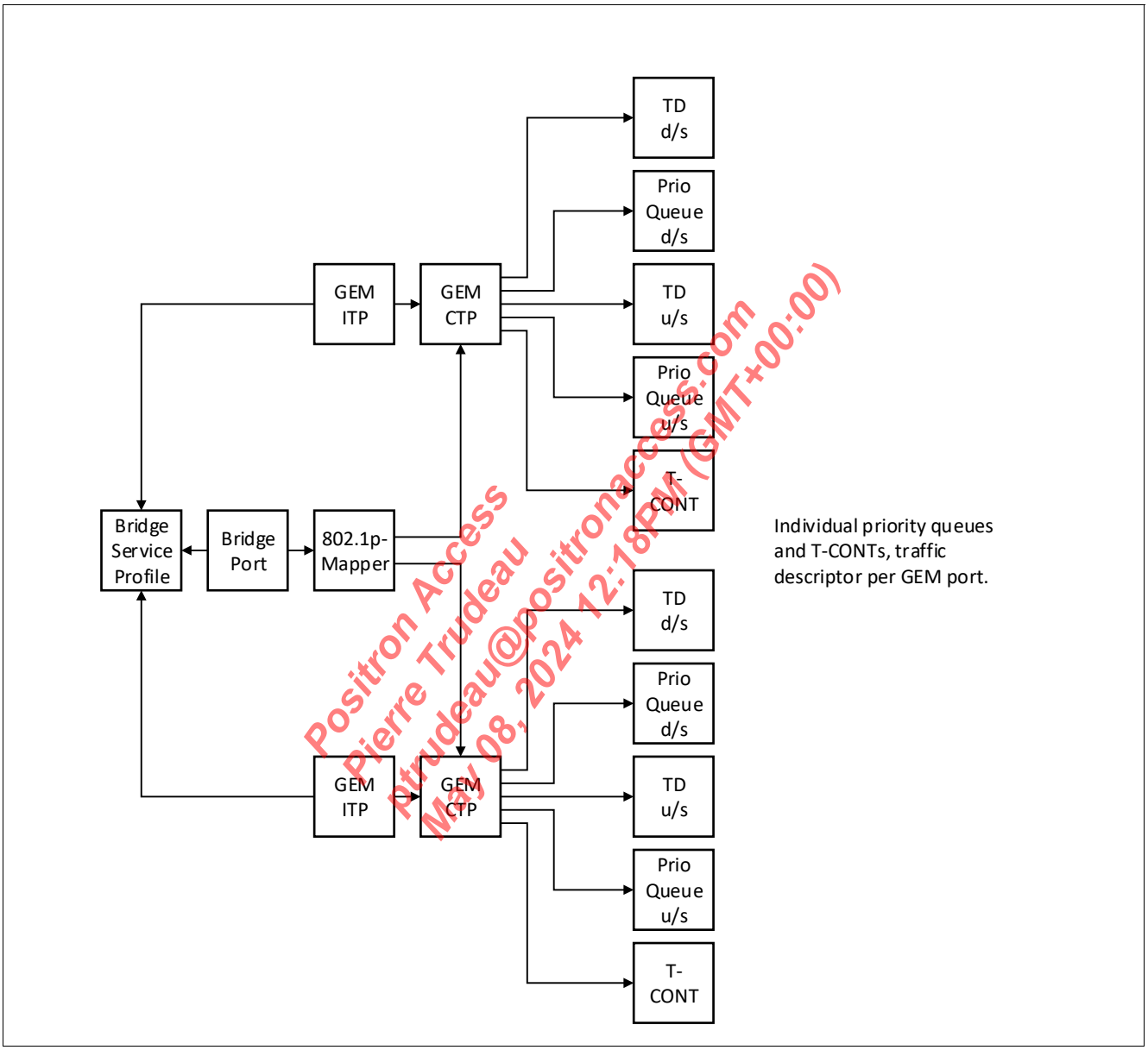


Figure 69 GEM CTP with Individual Priority Queues and T-CONTs

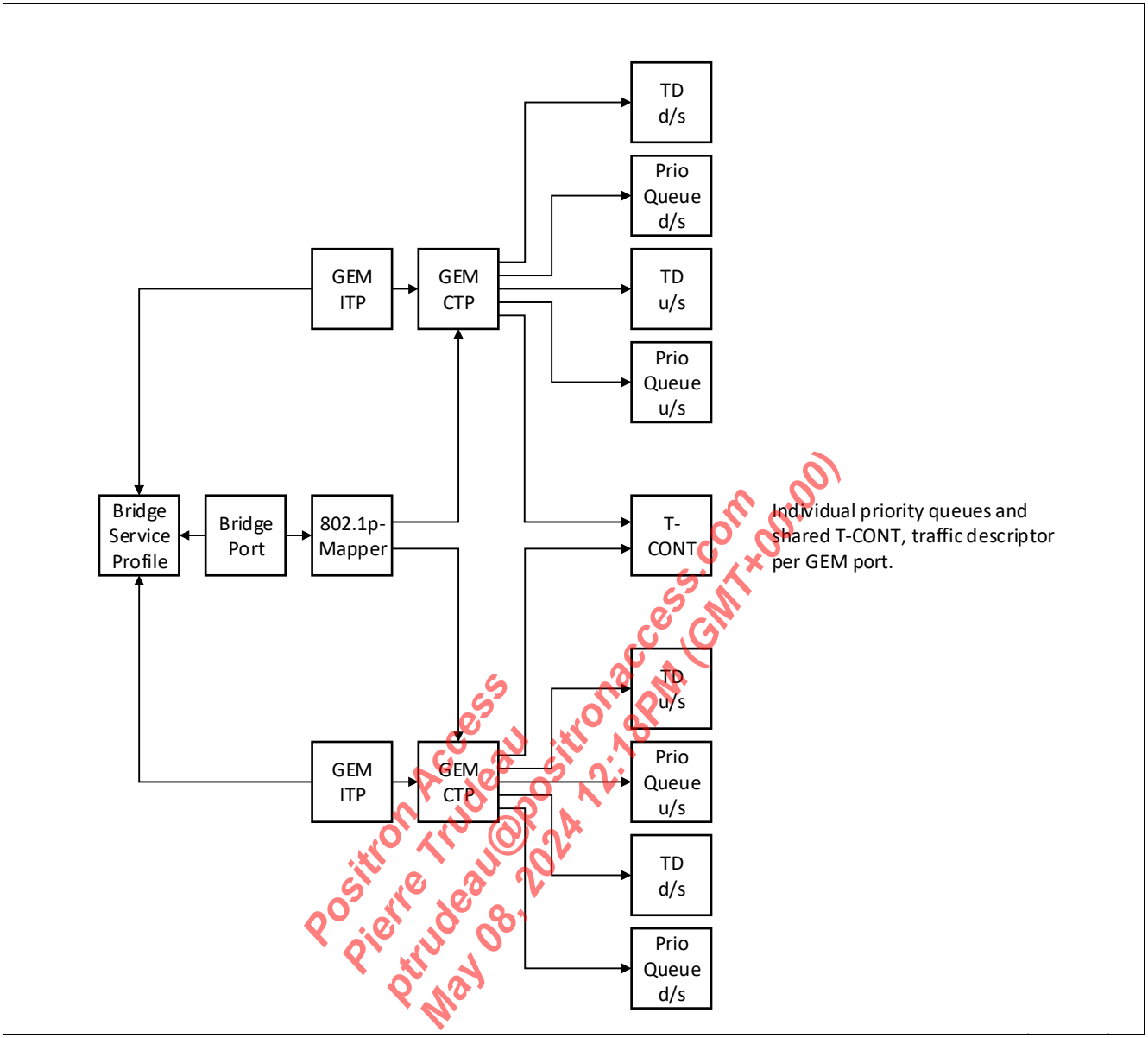


Figure 70 GEM CTP with Individual Priority Queues and Shared T-CONT

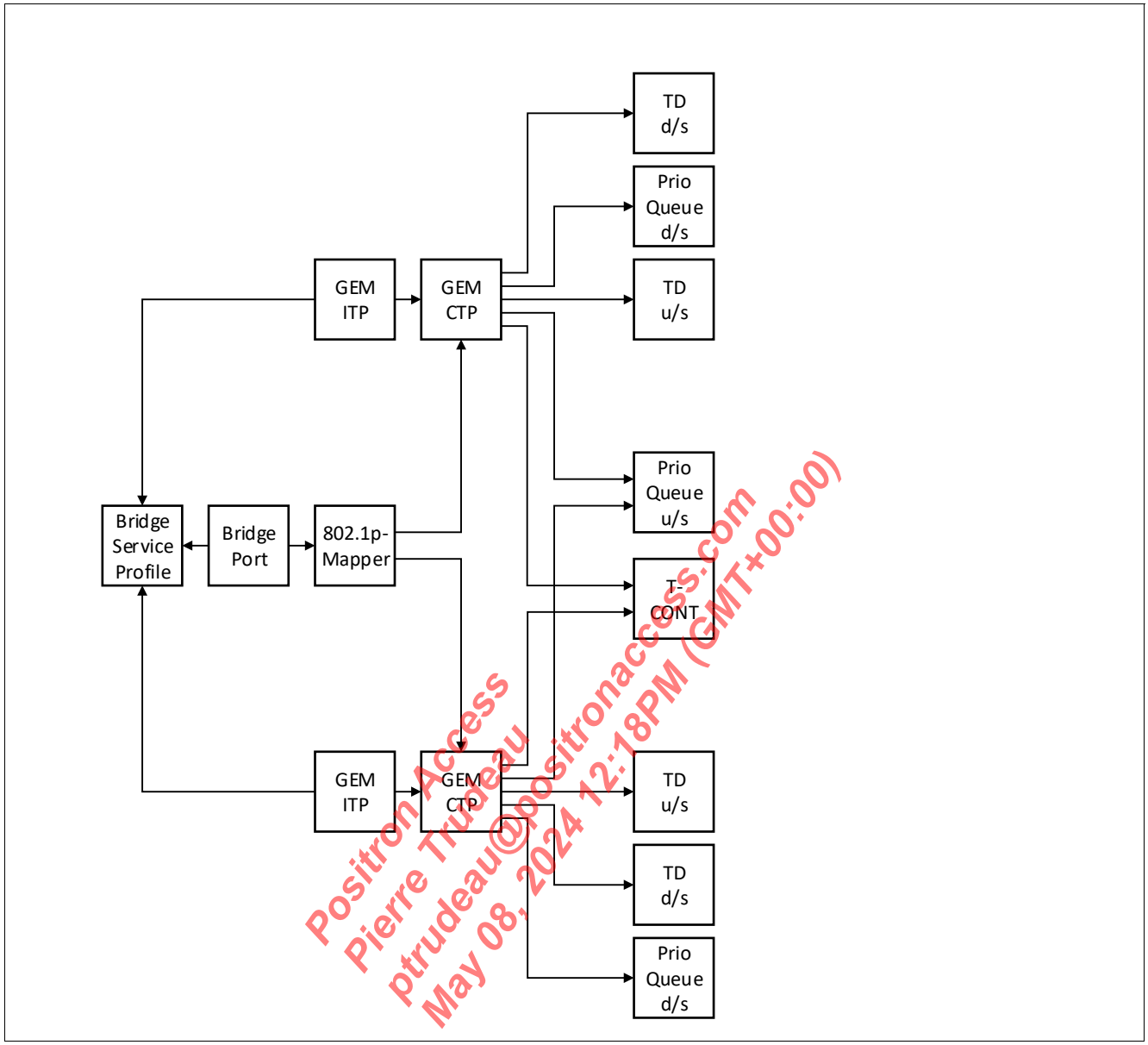


Figure 71 GEM CTP with Shared Priority Queues and Shared T-CONT

Figure 72 shows an example configuration for simple GEM CTP QoS setup sequence using the DP driver QoS API. Depending on the OMCI stack implementation, the function calls may follow a different order.

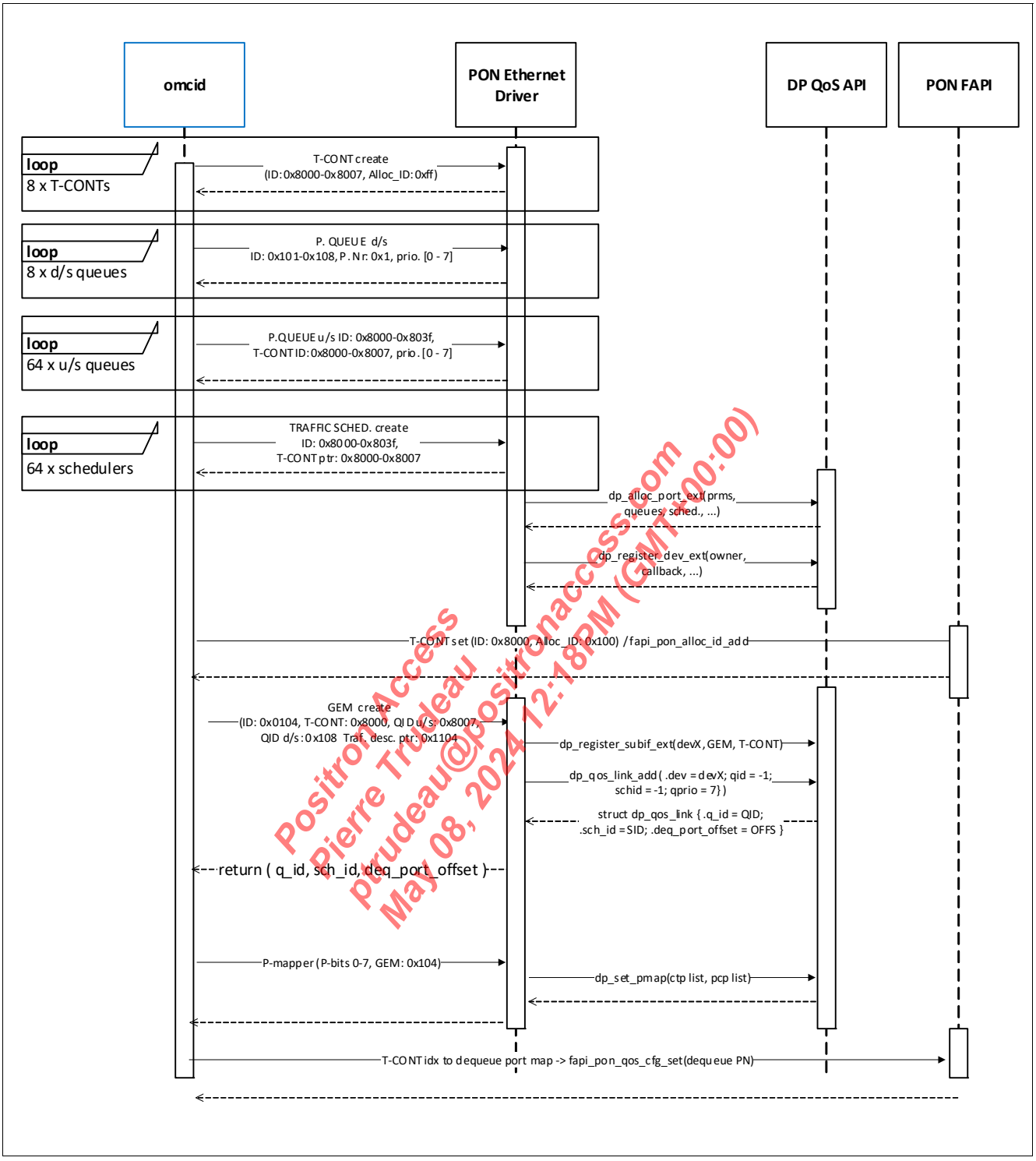


Figure 72 Possible OMCI QoS Configuration using DP Driver APIs over the PON Ethernet Driver

4.20.2.5 GEM Interworking Termination Point

ME class: 266

An instance of this managed entity is created at the request of the OLT and represents a point in the ONU where the interworking of a service or underlying physical infrastructure (here: Ethernet) to GEM layer takes place. At this point, the GEM packets are generated from an Ethernet bit stream or an Ethernet bit stream is reconstructed from the GEM packets.

Note: The relationship between the GEM ITP and a GEM port definition is as follows:
GEM ITP, GEM port network CTP connectivity pointer: points to the related GEM CTP.
GEM CTP, port ID: provides the 12-bit GEM/XGEM Port ID.

Managed Entity ID

2 bytes, read-only, set-by-create.

This attribute is a unique number to identify this managed entity, assigned by the OLT.
The OLT uses the same number value as for the related GEM/XGEM port ID.

GAL Loopback Configuration

1 byte, read, write.

This attribute is used to activate a GEM port loopback in the GPE module. The affected GEM/XGEM port ID is retrieved by checking the GEM port network CTP connectivity pointer to the attached **GEM Port Network CTP**, which holds the GEM/XGEM port ID (see **Port ID Value**).

Interworking Option, Service Profile Pointer, and Interworking Termination Point Pointer

1 byte, read, write, set-by-create.

Only interworking options 1 (bridging) and 6 (downstream broadcast) are supported per GEM port. Bridging that includes priority mapping in upstream direction is also reflected by code point 1.

Interworking Option 1 – Bridging or Bridging/Mapping

Interworking Option 5 – 802.1p Mapper without bridge

Note: This configuration is not supported.

Interworking Option 6 – Downstream Broadcast

GEM Port Network CTP Connectivity Pointer

2 bytes, read, write, set-by-create.

Each GEM port ITP must point to a single, individual GEM port network CTP. Multiple GEM ITP managed entities must not point to the same GEM port Network CTP.

GAL Profile Pointer

2 bytes, read, write, set-by-create.

This selects a **GAL Ethernet Profile** managed entity providing a single attribute, the maximum GEM frame size. Upon creation of this managed entity, the **Maximum GEM Payload Size – G.984 Operation Mode** of the related **GAL Ethernet Profile** is evaluated and configured for this GEM port by using `fapi_pon_gem_port_cfg_set` (`gem_max_size`).

When the pointer is changed, the hardware configuration is updated.

When the pointer is invalid (does not point to an existing **GAL Ethernet Profile**), the maximum size is set to a value of 4095 (no limitation).

PPTP Counter

1 byte, read-only.

This value reports the number of PPTP managed entity instances associated with this **GEM Interworking Termination Point**. No hardware programming is required for this attribute. The OMCI software scans all managed entities pointing to this one. Each time one of these pointers is added or removed, the PPTP counter value must be updated.

Relevant managed entities pointing to a **GEM Interworking Termination Point** are:

- **PPTP Ethernet UNI**

Operational State

1 byte, read-only.

The operational state of the GEM port is checked by calling `fapi_pon_gem_port_id_get` and checking the status of `is_downstream` and `is_upstream`. When one of these values is other than 0, the operational state is enabled (0).

```
operational_state = !(is_downstream || is_upstream)
```

Other Programming

When a GEM ITP managed entity is directly connected with a **MAC Bridge Service Profile** managed entity, the configuration software must autonomously insert a bridge port in between. There is no way to connect a MAC bridge directly to a GEM ITP by hardware/firmware configuration.

The port ID assignment in upstream direction is done either in:

- the bridge port table, when the interworking mode is bridge port, or
- the 802.1p mapper table, when the interworking mode is p-mapper.

When the interworking option is 1 (MAC bridge LAN), the service profile pointer is directed to a **MAC Bridge Service Profile** managed entity. In this case, an additional MAC Bridge Port must be created whose bridge ID points to the same MAC bridge service profile managed entity and whose traffic profile pointer points to this GEM interworking termination point.

Services other than MAC bridged LAN or 802.1 p mapper are not supported.

Actions

Create, delete, get, set

AVC Notifications

AVC #6: Operational state change

Note: The usage of this AVC is optional.
See #ifdef INCLUDE_OMCI_ONU_GEM_IWTP_AVC in the source code of omci_gem_interworking_tp.c.

Alarm Notifications

There are no alarm notifications defined for this managed entity.

4.20.2.6 Multicast GEM Interworking Termination Point

ME class: 281

This managed entity is created at the request of the OLT and is mostly identical to the **GEM Interworking Termination Point**, with two differences:

- The **Service Profile Pointer** is not used.
- The interworking termination point pointer is not used.
- The **PPTP Counter** is not used.
- The **GAL Profile Pointer** is not used.
- The loopback function for multicast GEM/XGEM port IDs is not possible.
- An additional multicast address table is provided.

Managed Entity ID

2 bytes, read-only, set-by-create.

This attribute is a unique number to identify this managed entity, assigned by the OLT.
The OLT uses the same number value as for the related GEM/XGEM port ID.

Interworking Option

1 byte, read, write, set-by-create.

Only interworking option 1 (bridging) is supported per GEM port. Other code points except for 0 (don't care) must be rejected.

IPv4 Multicast Address Table

12 x N bytes, read, write.

The multicast address table is held in software to allow or deny the additions of IPv4 multicast streams through the IGMP control messages. When an IGMP join message is snooped by the software, the related hardware configuration is set only when the desired multicast stream fits the definition given by the multicast address table.

IPv6 Multicast Address Table

24 x N bytes, read, write.

The multicast address table is held in software to allow or deny the additions of IPv6 multicast streams through the MLD control messages. When an MLD join message is snooped by the software, the related hardware configuration is set only when the desired multicast stream fits the definition given by the multicast address table.

Service Profile Pointer

2 bytes, read, write, set-by-create.

This attribute is set to 0 and not used. For backward compatibility, it may also be set by the OLT to point to a MAC bridge service profile or 802.1p mapper service profile.

PPTP Counter

1 byte, read-only.

This attribute represents the number of instances of PPTP managed entities associated with this instance of the **Multicast GEM Interworking Termination Point**.

This attribute does not convey any information that is not available elsewhere. It may be set to FF_H and not used by the OLT.

Relevant managed entities pointing to a **Multicast GEM Interworking Termination Point** are:

- **PPTP Ethernet UNI**

Operational State

1 byte, read-only.

This attribute indicates whether or not the managed entity is capable of performing its function.

Valid values are enabled (0) and disabled (1).

GAL Profile Pointer

2 bytes, read, write, set-by-create.

This attribute is set to 0 and not used by the OLT. It is ignored by the software.

For backward compatibility, it may also be set to point to a **GAL Ethernet Profile**.

Actions

Create, delete, get, get next, set

AVC Notifications

AVC #6: Operational state change

Alarm Notifications

There are no alarm notifications defined for this managed entity.

4.20.2.7 GEM Port Performance Monitoring History Data

ME class: 267

This managed entity is created at the request of the OLT and provides counter information that is related to a selected GEM Port (when possible) or to the global downstream GEM traffic.

*Note: This is a deprecated managed entity and has been replaced by the **GEM Port Network CTP Performance Monitoring History Data** in new applications. It is described here for reference only.*

4.20.2.8 GAL Ethernet Profile

ME class: 272

This managed entity is created at the request of the OLT.

There is a single configuration value provided by this managed entity, which is the maximum GEM payload size related to a certain GEM port. It is possible for multiple GEM ports to be linked to a single GAL Ethernet profile.

The PON receive hardware is not limited in terms of GEM payload size and supports any GEM frame size up to the limit defined by the related standard of operation. Hence, this parameter has no influence on the downstream GEM frame reception.

In upstream direction, it is possible to set a hardware limit. When the Ethernet or OMCI frame to be transmitted is larger, it is fragmented into multiple GEM/XGEM frames.

Managed Entity ID

2 bytes, read-only, set-by-create.

The OLT uses an arbitrary number to identify one or more of these managed entities, starting at 1.

Maximum GEM Payload Size – All Operation Modes

2 bytes, read, write, set-by-create.

The implementation must take care of the creation and update order of the related **GEM Interworking Termination Point** managed entities, pointing to this profile (using the **GAL Profile Pointer**).

Each time the value of this attribute is set or changed, the related GEM ports must be identified and all of them must be (re-)configured.

The valid range of this value received in this attribute depends on the operation mode and is used to configure the PON IP upstream path through `fapi_pon_gem_port_cfg_set` using the value of `gem_max_size`.

Note: To avoid excess packet overhead, the default software implementation does not accept values smaller than 2048 bytes and silently sets this value when the OLT tries to assign a lower value. This lower limit is controlled by a #define value in the source code and is modifiable to fit other application requirements.

Maximum GEM Payload Size – G.984 Operation Mode

The valid range of this parameter is from 0 to 4096 byte.

The hardware configuration is limited to a granularity of 1 byte for the only upstream rate of 1.24416 Gbit/s. When reading back the configured value, the PON library function returns the configured value.

Maximum GEM Payload Size – G.987/9807/989 Operation Mode

The valid range of this parameter is from 8 to 16380 byte.

The hardware configuration is limited to a granularity of 4 bytes for upstream rates of 2.48832 Gbit/s and 9.95382 Gbit/s. When reading back the configured value, the PON library function returns the ceiling value of what has been configured.

Pseudo code performed by the firmware:

```
gem_max_size_used = 4 * ceil(gem_max_size_configured/4);
```

Actions

Create, delete, get, set

Notifications

There are no alarm or AVC notifications defined for this managed entity.

4.20.2.9 GAL Ethernet Performance Monitoring History Data

ME class: 276

This managed entity is created at the request of the OLT and represents a single counter that accumulates the number of lost downstream GEM frames that are related to a dedicated GEM Port.

Note: This managed entity is implemented as a template only.

Managed Entity ID

2 bytes, read-only

The same number is used as for the **GEM Interworking Termination Point**, to which this managed entity is related.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** managed entity that is pointed to by this value provides one alarm threshold which is used for the Discarded Frames. The **Threshold Data 2** managed entity is not used.

Discarded Downstream Frames

4 bytes, read-only

This counter is not implemented in the reference software.

Discarded Upstream Frames

4 bytes, read-only

The hardware counter is read from the GSWIP hardware, for an implementation example see `omci_gal_ethernet_pmhd.c`.

Actions

Create, delete, get, get current data, set

Threshold Crossing Alerts

The following Threshold Crossing Alerts (TCA) are sent:

TCA #0: Discarded downstream frames

TCA #1: Discarded upstream frames

4.20.2.10 FEC Performance Monitoring History Data

ME class: 312

This managed entity is created at the request of the OLT and represents a single set of counters that accumulates the number of downstream FEC-related events. If the downstream FEC mode is disabled, then all counters deliver a value of 0.

The hardware counters are read regularly and accumulated in a set of 64-bit software counter variables. To read the counter values use the PON adapter function `pa_fec_pmhd_ops`.

Please see the **Enhanced FEC Performance Monitoring History Data** in **Chapter 4.20.10.1** for the use in XG-PON, XGS-PON, and NG-PON2 operation modes.

Note: In 10G PON operation modes (XG-PON, XGS-PON, NG-PON2) the Total Words counter overflows within a 15-minute interval.

Readout of the FEC Operation Mode

The downstream FEC operation mode is selected by the OLT through the (X)G-PON hardware protocol. The FEC operation can be read through `fapi_pon_gpon_status_get` (`fec_status_ds`).

Managed Entity ID

2 bytes, read-only, set-by-create

In G-PON, XG-PON, and XGS-PON operation mode this is the same number as used for the **ANI-G**, to which this managed entity is related. In NG-PON2 operation mode this is the same number as used for the related **TWDM Channel** managed entity.

Note: In NG-PON2 operation mode, this managed entity cannot be connected to the ANI-G but only to a TWDM Channel managed entity.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** managed entity that is pointed to by this value provides three alarm thresholds which are used for the Corrected Bytes, Corrected Code Words, Uncorrectable Code Words, and FEC Seconds. The **Threshold Data 2** managed entity is not used.

Corrected Bytes

4 bytes, read-only

Corrected bytes are counted by a hardware counter in the PON IP module and accumulated in a software counter by using the PON adapter function `pa_fec_pmhd_ops (cnt_corrected_bytes)`.

Corrected Code Words

4 bytes, read-only

Corrected code words are counted by a hardware counter in the PON IP module and accumulated in a software counter by using the PON adapter function `pa_fec_pmhd_ops (cnt_corrected_code_words)`.

Uncorrectable Code Words

4 bytes, read-only

Uncorrectable code words are counted by a hardware counter in the PON IP module and accumulated in a software counter by using the PON adapter function `pa_fec_pmhd_ops (cnt_uncorrected_code_words)`.

Total Code Words

4 bytes, read-only

The total number of received FEC code words (FEC blocks) is counted by a hardware counter in the PON IP module and accumulated in a software counter by using the PON adapter function `pa_fec_pmhd_ops (cnt_total_code_words)`.

FEC Errored Seconds

2 bytes, read-only

This counts the number of one-second intervals in which at least one uncorrectable FEC codeword was seen.

Within the 15-minute time interval, a maximum increment of 900 can be seen.
Check `pa_fec_pmhd_ops (cnt_fec_seconds)` to retrieve the value.

Actions

Create, delete, get, set

Attribute Value Change Notifications

There are no Attribute Value Changes defined for this managed entity.

Alarm Notifications

These notifications (Threshold Crossing Alerts) are sent by this managed entity:

- Alarm #0: Corrected bytes
- Alarm #1: Corrected code words
- Alarm #2: Uncorrectable code words
- Alarm #4: FEC seconds

4.20.2.11 Priority Queue

ME class: 277

A set of priority queues is created by the ONU upon OMCI startup and offered to the OLT.

Each of these managed entities represents a single egress[1)] queue. An upstream hardware queue must be connected to a hardware traffic scheduler. Direct connection of a hardware queue to a T-CONT is not supported by the hardware.

To provide the flexible **T-CONT** properties from the OMCI point of view, hardware scheduler resources must be attached to a hardware T-CONT to form a logical OMCI T-CONT. This is depicted in **Figure 73** and **Figure 74**. Using a single-stage scheduling supports up to eight queues for a **T-CONT**, hierarchical scheduling is used to provide more than eight queues or mixed priority handling methods.

The hardware schedulers used to form the logical OMCI T-CONTs are already taken from the overall pool of hardware schedulers and are not available as OMCI scheduler managed entities.

Figure 73 shows the connection of eight queues, handled according to a strict priority or weighted round robin method. In strict priority mode, the selected scheduler input number defines the priority of a queue.

Figure 74 shows the connection of six queues to a **T-CONT**, with two of them using strict priority while the remaining four are served in a weighted round robin scheme.

When implementing hardware scheduler resources in the **T-CONT** managed entity, it must be taken into account that additional traffic scheduler managed entities in the OMCI modes contribute to the overall level of scheduling.

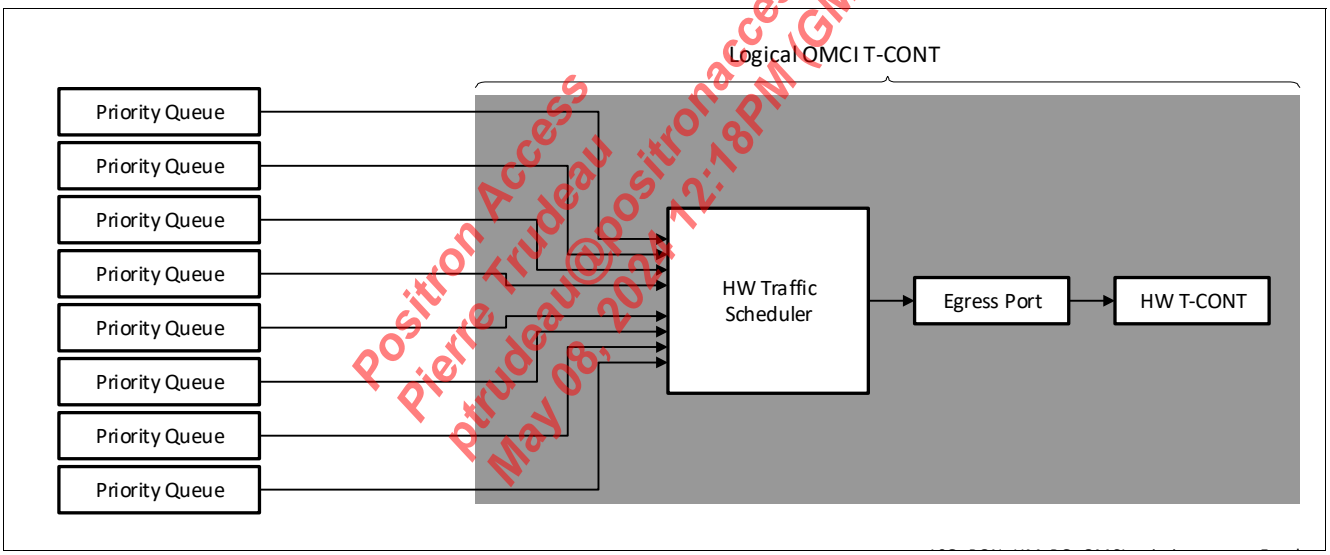


Figure 73 Logical T-CONT with Multiple Queues and Single-stage Scheduling

1) While OMCI uses the expression “priority queue”, the software driver partly uses “egress queue” for these.

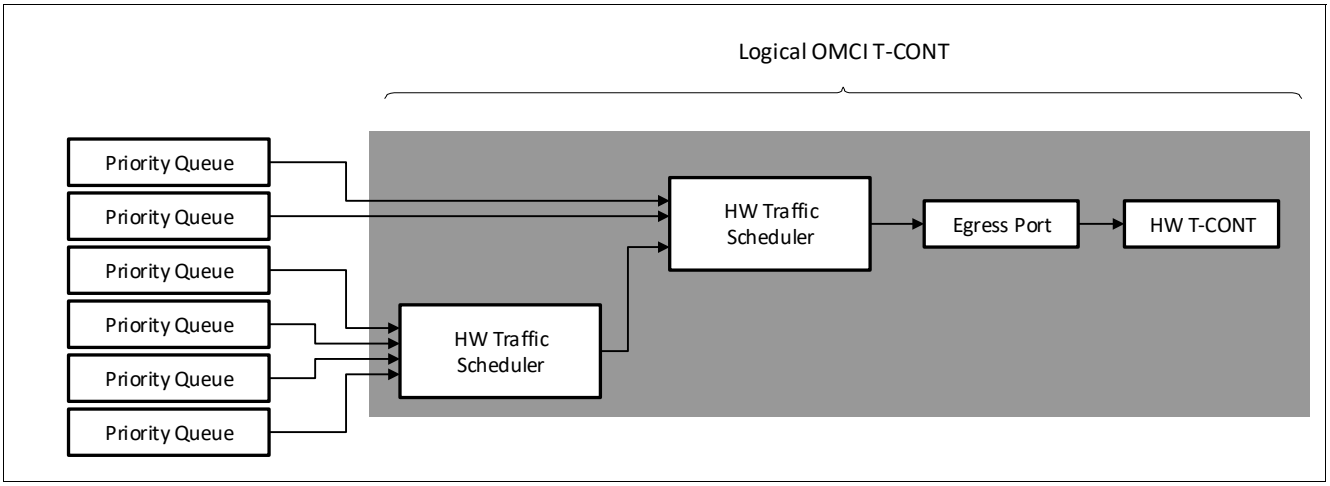


Figure 74 Logical T-CONT with Multiple Queues and Hierarchical Scheduling

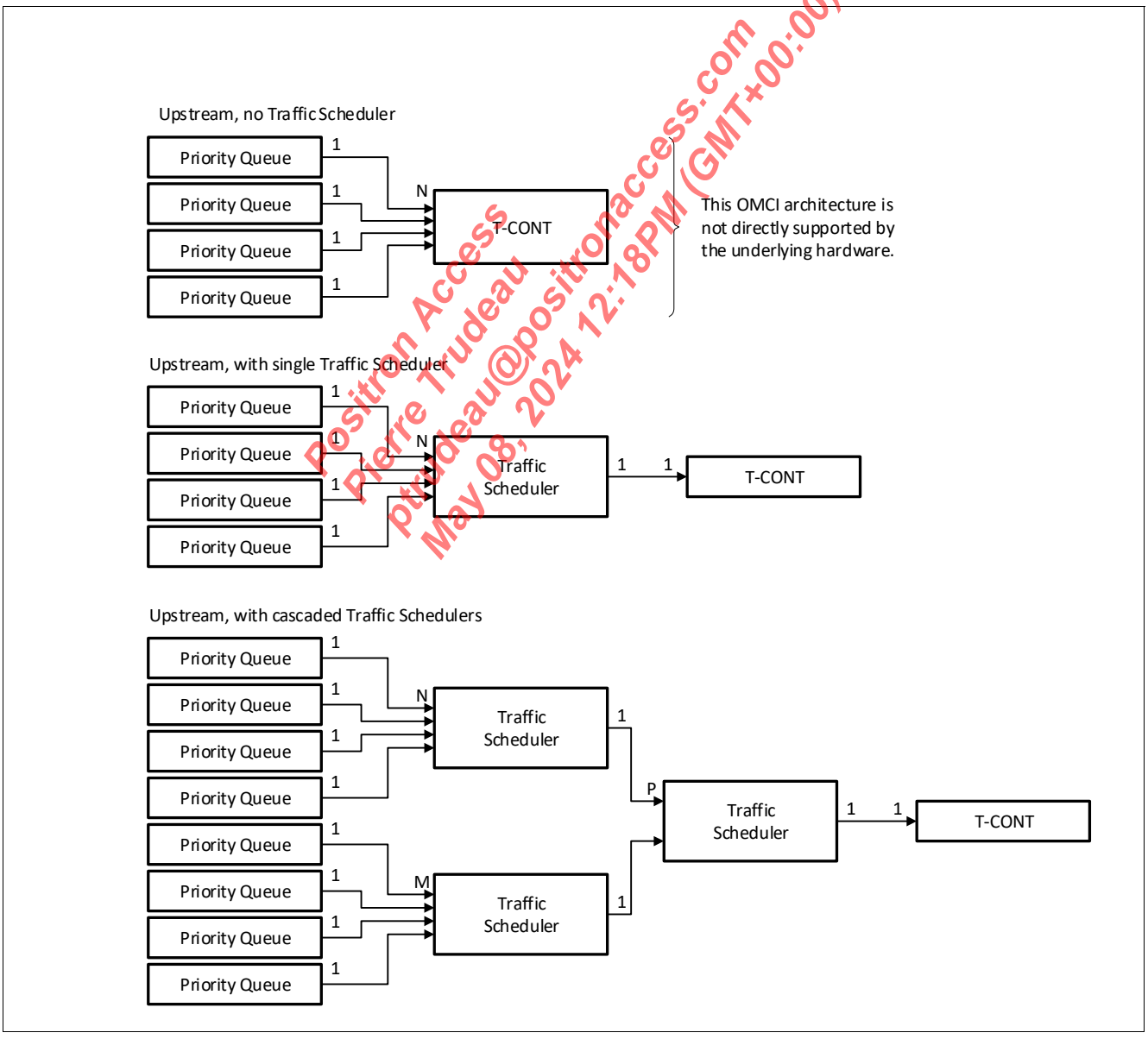


Figure 75 Priority Queue OMCI Configuration Options – Upstream

Managed Entity ID

2 bytes, read-only

The managed entity ID represents the queue identification number. The MSB defines the queues direction with:

- 0000_H … $7FFF_H$: downstream (used: 32 instances, 8 per UNI/VEIP)
- 8000_H … $FFFF_H$: upstream (used: 128 instances, 4 per T-CONT, up to 32 T-CONTs)

The lower 15 bits define the queue ID, this must be mapped to the physical queue number of 0 to the maximum number of available queues.

The managed entity ID values are assigned in the MIB initialization file.

Queue Configuration Option

1 byte, read-only

This parameter is defined by the application and is not configurable by the OLT.
The setting is provided in the MIB initialization file:

- 0: The individual space per queue is reported.
- 1: The overall memory space for queuing is reported.

Maximum Queue Size

2 bytes, read-only

This attribute specifies the amount of memory available either for the sum of all queues (when the queue configuration option is 1) or for the specific queue (when the queue configuration option is 0). This limit is defined by the application and reported to the OLT through this value.

Queue Configuration Option 0

The hardware supports a fraction of the shared queue size.
The value reported by this attribute is defined by the total number of bytes available divided by the priority queue scale factor as given by the ONU2-G managed entity and by the number of queues used. The same value is reported for each of the queues.

Queue Configuration Option 1

The hardware supports a total shared queue size.
The value reported by this attribute is defined by the total number of bytes available divided by the priority queue scale factor as given by the ONU2-G managed entity. When it does not support this attribute, the factor is set to 48.
The queue size must be configurable using the `ip` command. This configuration is done on the GEM network devices as the queues are always associated with GEM ports:

```
ip link set gem123 txqueuelen [length]
```

The length is given in number of packets to convert it to bytes. The value configured must be divided by the network device MTU size.

Note: With a `priority_queue_scaling_factor` *of 1, the maximum queue size to be reported is 65536 byte. This is less than the hardware is able to provide. To enable larger queue sizes, the* `priority_queue_scaling_factor` *must be increased.*

Allocated Queue Size

2 bytes, read, write

The allocated queue size is physically defined by the maximum value given through the configuration of the individual **Packet Drop Queue Thresholds**. This maximum value is reported upon an attribute read. Writing to this attribute is ignored, it is written as 0. To modify the allocated queue size, the drop thresholds must be adapted. When queue configuration option 0 is selected, it is possible to set the drop thresholds such that the reported allocated queue size is larger that the reported maximum queue size. This overbooking is allowed and supported by the hardware.

Note: When an allocated queue size different from 0 but smaller than given by the maximum drop threshold is received, this must be ignored. A warning is issued, when the debug output is enabled.

Discard-block Counter Reset Interval

2 bytes, read, write

This attribute defines the time interval used to clear the software-based counter that accumulates the number of discarded blocks. No hardware setting is affected by this value.

This function is currently not implemented.

Threshold Value for Discarded Blocks due to Buffer Overflow

2 bytes, read, write

This attribute defines the threshold applied to generate a block loss alarm. It is compared with the discard-block counter value and an alarm is triggered when the counter value exceeds this threshold. The counter values are compared against the threshold in regular intervals.

Note: The hardware does not support the counting of discarded bytes/blocks but only the number of discarded packets. An estimation for an average number of bytes per packet must be used to set the threshold.

Related Port

4 bytes, read, write

The first two bytes of this attribute link the queue to an upstream ANI port and T-CONT, a downstream UNI port, or to a CPU host port (virtual Ethernet interface point, VEIP). The hardware programming depends on the queue location (upstream or downstream), which is derived from the managed entity ID as previously described.
The second two bytes define the queue priority in a range from 0x0000 (highest) to 0x0FFF (lowest).

Traffic Scheduler Pointer – Upstream

2 bytes, read, write

This attribute links the queue to a traffic scheduler. Resource management is required to administrate the available number of traffic schedulers. When the queue is directly connected to a T-CONT, this pointer is set to 0 (default value).

Traffic Scheduler Pointer – Downstream

2 bytes, read, write

The downstream queues use a fixed configuration to assign a set of queues to a certain UNI port. This is defined in the MIB initialization file and not controlled by the OLT.

Weight

1 byte, read, write

While the OMCI attribute allows for values from 0 (low) to 255 (high), this must be mapped to a valid range as provided by the hardware.

Back Pressure Operation

2 bytes, read, write

The back pressure operation is not supported per queue, any other setting than the default value (1 = no backpressure) is ignored.
Although the nominal default for this value is 0, it is set to 1 for all queues in the MIB initialization file.

Back Pressure Time

4 bytes, read, write

The back pressure operation is not supported per queue, any value different from the default value of 0 is ignored.

Back Pressure Occur Queue Threshold

2 bytes, read, write

The back pressure operation is not supported per queue, any value different than 0 is ignored.

Back Pressure Clear Queue Threshold

2 bytes, read, write

The back pressure operation is not supported per queue, any value different than 0 is ignored.

Packet Drop Queue Thresholds

8 bytes, read, write

This attribute provides four separate values, which are:

- Minimum threshold for green packets
- Maximum threshold for green packets
- Minimum threshold for yellow packets
- Maximum threshold for yellow packets

The unit for all thresholds is variable. The factor is given by the **GEM Bock Length**, as defined by the ANI-G managed entity attribute (48-byte units by default) for implementations that follow ITU-T G.984.4 [52]. When the ONU implements ITU-T G.988 [56], the factor is given by the **Priority Queue Scale Factor** attribute of the **ONU2-G** managed entity (1-byte units by default).

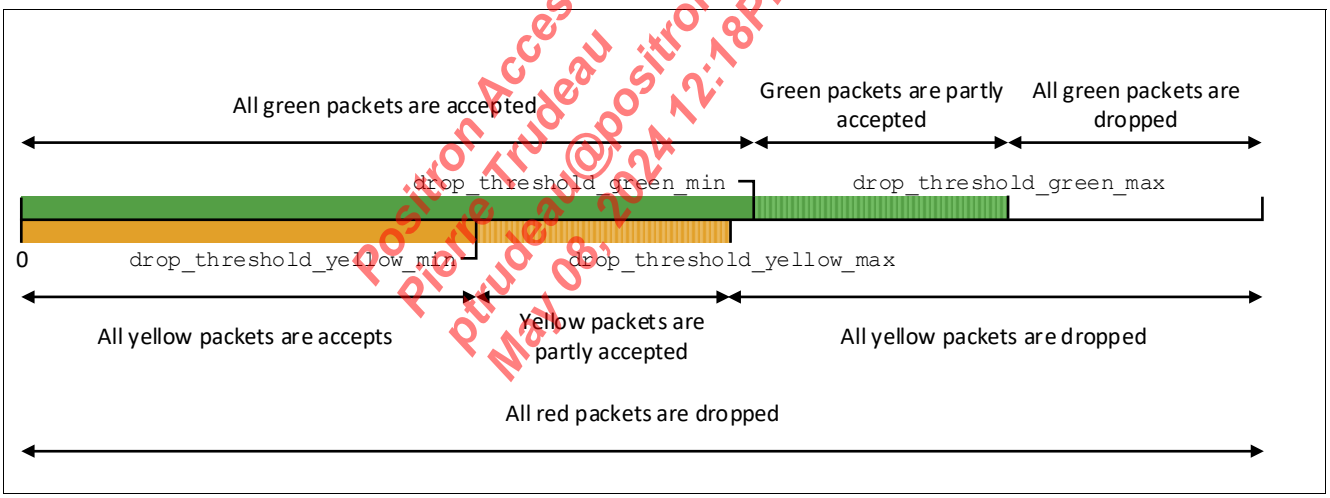


Figure 76 Priority Queue Threshold Parameter Definitions

Packet Drop Maximum Probability (max_p)

2 bytes, read, write

This attribute provides two separate values, the drop probability for green and yellow packets.

Together with the **Packet Drop Queue Thresholds** attribute, this defines the queue acceptance characteristic.

Figure 77 depicts the definition of the attribute values and their basic dependencies.

Figure 78 shows an example configuration.

As a default configuration, it is recommended to set the drop probability values to 0.

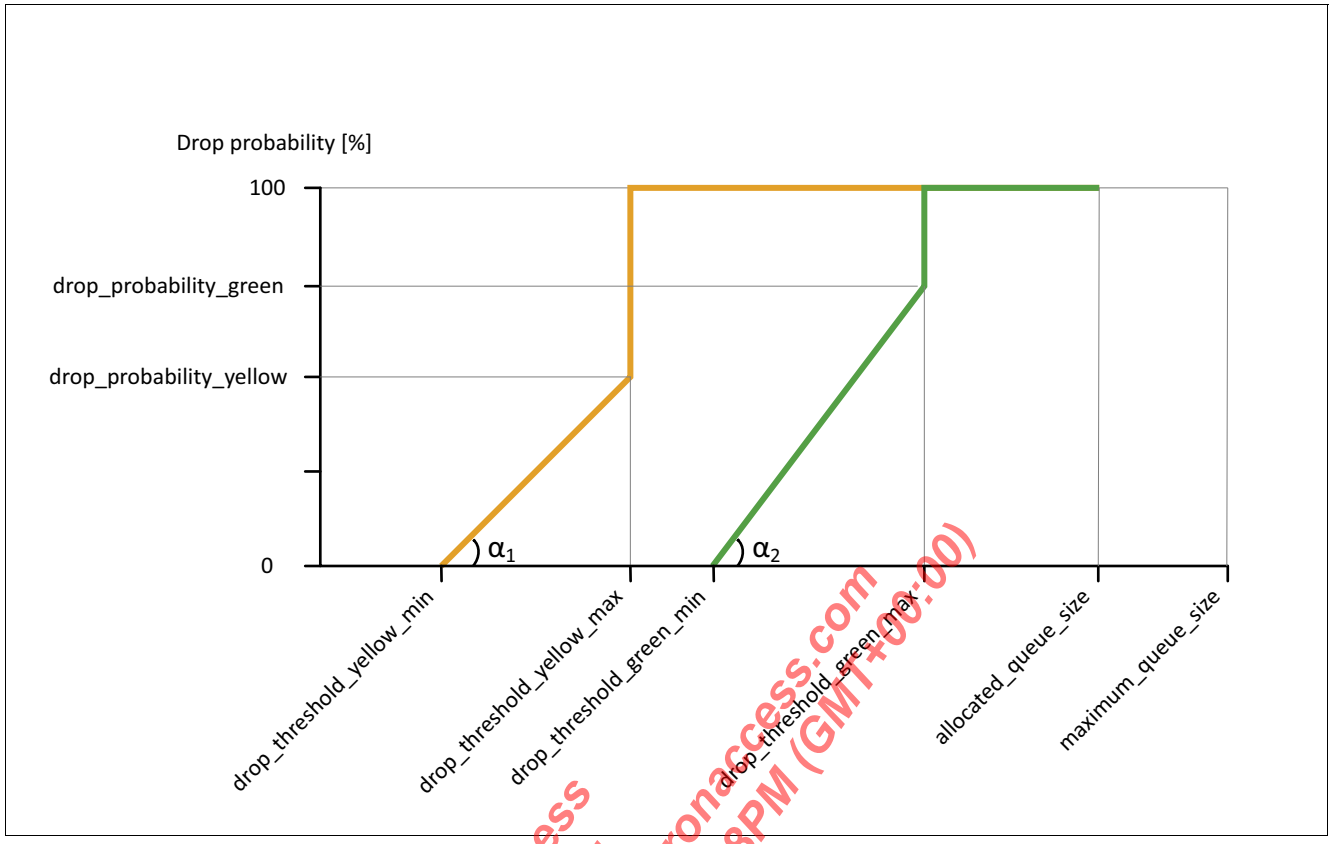


Figure 77 Priority Queue Acceptance Parameter Definitions

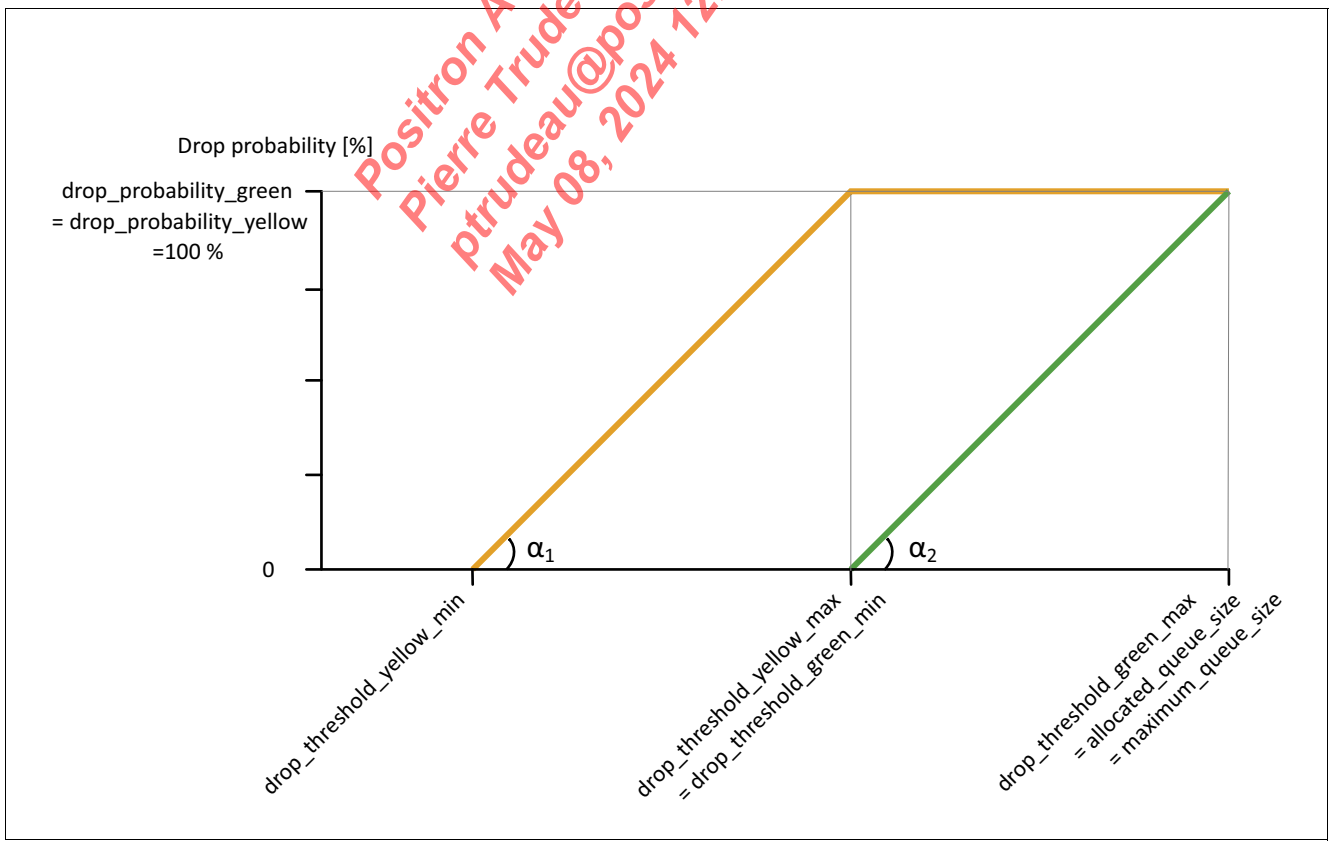


Figure 78 Priority Queue Acceptance Example Configuration

The slope in percent is calculated using the formula:

$$\alpha = \frac{drop\_\max\_probability}{drop\_threshold\_\max - drop\_threshold\_\min} \times 100 \quad (1)$$

The probability value is already converted by adding one to the original value and dividing by 256.

The configuration of the WRED parameters is done by using `tc-red` qdiscs. The first one provides the green packets thresholds and max probability and the second the values for the yellow packets. For example:

```
tc qdisc add dev ens6 root handle 11: red limit 100k min [gr_min] max [gr_max] avpkt
1k burst 55 probability [gr_probabililty]
tc qdisc add dev ens6 parent 11: handle 12: red limit 100k min [yl_min] max [il_max]
avpkt 1k burst 55 probability [gr_probabillity]
```

Queue Drop Averaging Coefficient (w_q)

1 byte, read, write

This attribute provides the weight factor used to calculate the average queue occupation. This a parameter common for all queues and configured during QoS engine initialization.
It must be modifiable by some means device tree, sysfs or netlink.

Drop Precedence Color Marking

1 byte, read, write

This attribute defines how color marking is performed on packets stored in the queue.
The packets entering the priority queues must be colored according to the specified algorithm. This is configured by using the `tc police qdisc` **[29]** and the newly introduced `tc-colmark` action.

This is an example on how a drop precedence color marking algorithm PCP_6P2D is configured:

```
tc filter add dev gem100 ingress flower action colmark marker pcp_6p2d
```

This GEM network device corresponding to the queue is being used in the filter command to setup the color marker type for the queue.

Actions

Get, set

Alarm Notifications

This alarm is defined for this managed entity:

- Block loss
  - This alarm is triggered when the number of lost blocks exceeds the threshold given by the **Threshold Value for Discarded Blocks due to Buffer Overflow** attribute. It is cleared with the next expiration of the **Discard-block Counter Reset Interval**.

This function must be implemented by the OMCI software. A driver event is not available to indicate a block loss. The upstream priority queue configuration is done by the Linux base network interface provided by the data path drivers or by using the QoS library for the downstream queues.

This alarm is not covered by the reference implementation.

Linux Implementation of Priority Queues

Figure 79 shows the mapping of the OMCI priority queues and schedulers to Linux network devices and qdisc. The priority queue and scheduler parameters must be configured by using the Linux `qdisc` hardware offloading support. The actual offloading must be implemented in the data path driver. This is done by implementing the network device operation hook `ndo_setup_tc`.

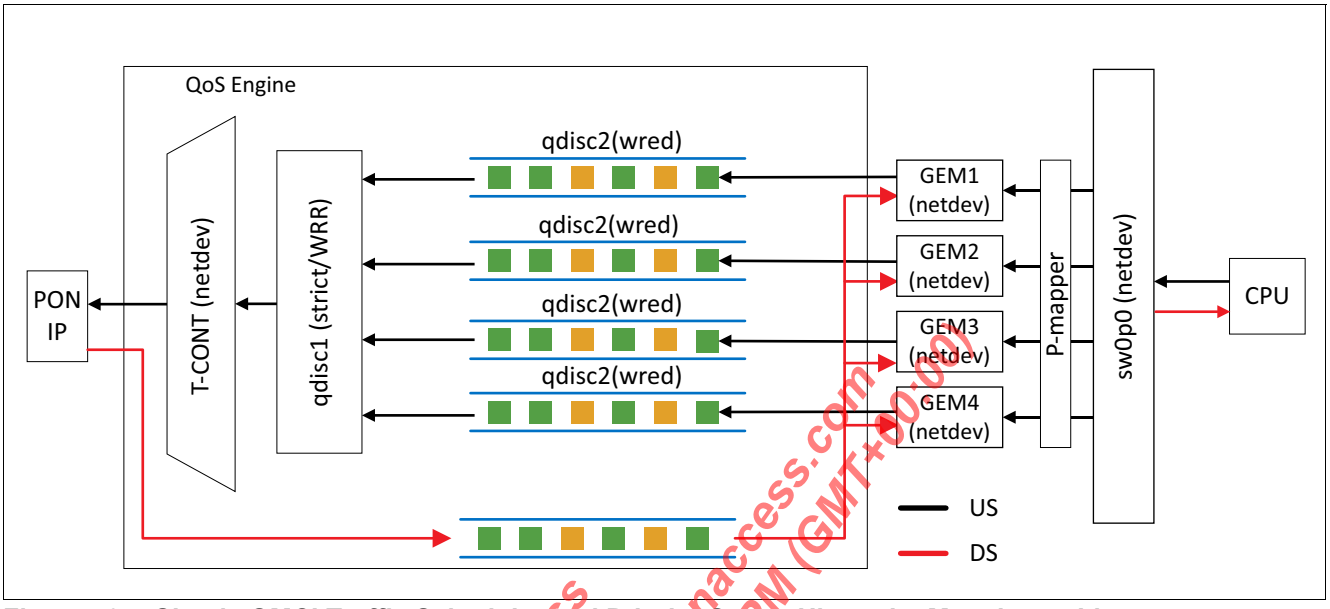


Figure 79 Simple OMCI Traffic Scheduler and Priority Queue Hierarchy Mapping to Linux

Figure 80 presents the traffic scheduler and priority queue configuration. This configuration is triggered by the OMCI stack for each GEM Port Network CTP with pointers to the priority queues, scheduler and T-CONTs it is assigned to.

The OMCI stack must trace the pointers to the corresponding MEs and get the configuration parameters for the dequeue port (base on the T-CONT index), scheduler and priority queues. Then, it must setup the dequeue port for the upstream traffic, configure the traffic scheduler policy and number of queues served and finally configure the queue parameters, such as WRED. In Figure 80, these are steps 1 to 3. The figure also shows one possible relationship between the tree MEs. It is possible to have a setup with only T-CONT and priority queue as T-CONT also has scheduling capabilities.

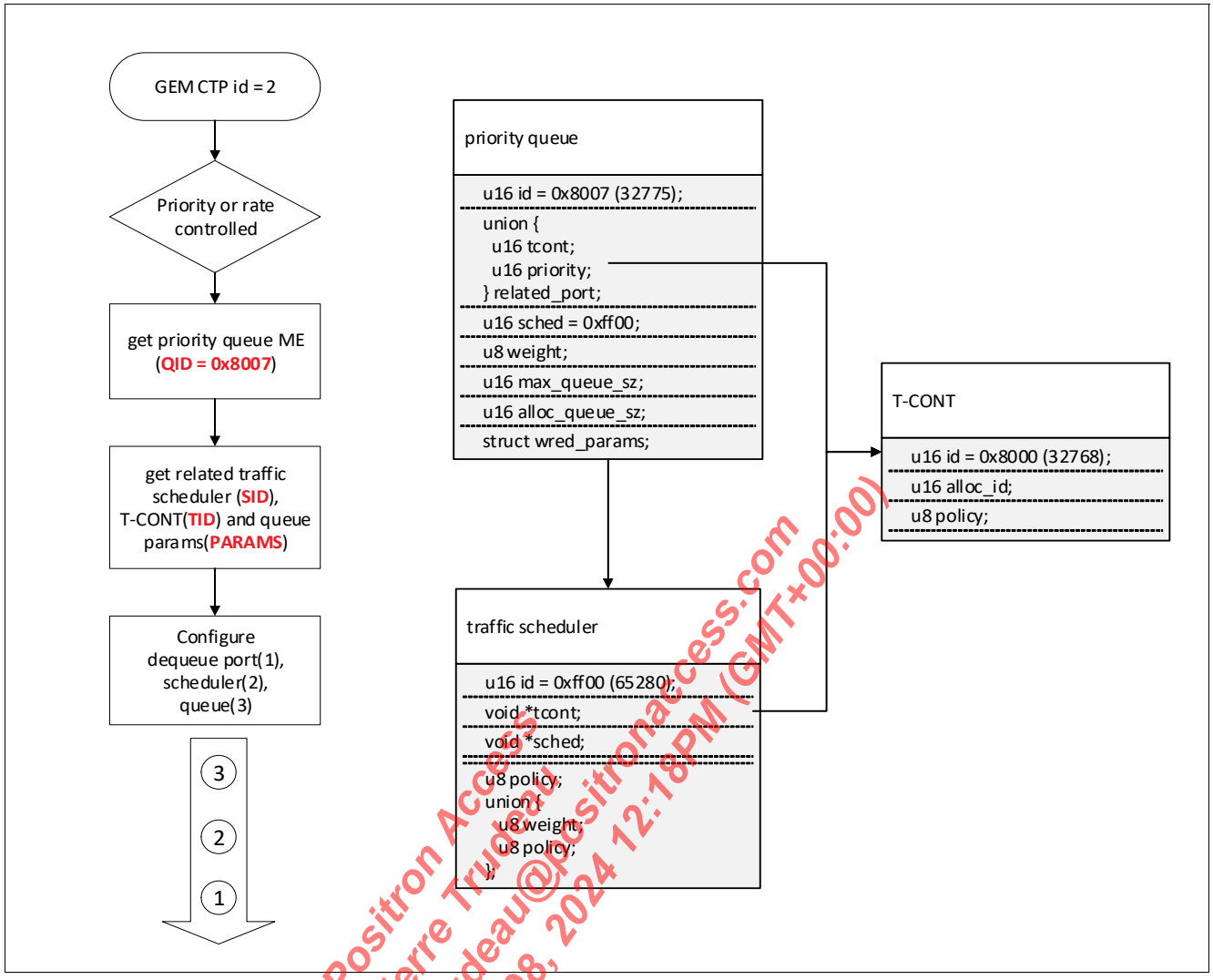


Figure 80 OMCI Stack Scheduler and Priority Queue Configuration Flow

*Note: For further details see **[11]**.*

4.20.2.12 Traffic Scheduler

ME class: 278

A set of traffic schedulers is created by the ONU upon OMCI startup. The list of available schedulers and their relationship to priority queues is defined in the MIB initialization file.

An instance of this managed entity represents a logical object that is able to control the upstream GEM/XGEM packets. A traffic scheduler accommodates the GEM/XGEM packets after a priority queue or another traffic scheduler managed entity and transfers them towards the next traffic scheduler or to a T-CONT.

Due to the limited number of hardware scheduler blocks, this managed entity is subject to resource management.

In addition to the OMCI-based usage of traffic schedulers, the hardware implementation uses traffic schedulers also in the downstream direction to connect the downstream queues with UNI ports. One scheduler and eight priority queues (egress queues) are used for each of the UNI ports in the downstream direction.

The scheduler configuration is done using the PON Ethernet driver Linux API. A Linux `qdisc` must be introduced to enable configuration of the scheduler policies and parameters. The `qdisc` must support the `ndo_setup_tc` network device management hook as in the case of priority queue configuration. See **Section 4.20.3.14.4 OMCI Extended VLAN Configuration in Linux** for details.

Managed Entity ID

2 bytes, read-only

This number indicates the physical capability that realizes the traffic scheduler. The first byte is the slot ID of the **Circuit Pack** with which this traffic scheduler is associated. For a traffic scheduler not associated with a circuit pack, the first byte is 0xFF. The second byte is the traffic scheduler ID, assigned by the ONU itself through the MIB initialization file. The traffic schedulers are numbered in ascending order, ranging from 0x00 to 0xFF in each circuit pack or in the ONU core.

T-CONT Pointer

2 bytes, read, write.

This attribute provides the upstream connection of the traffic scheduler to a **T-CONT**. The physical scheduler output number (dequeue port) must be configured to the T-CONT index value. This is done by the PON Ethernet driver when creating the T-CONT network device.

Traffic Scheduler Pointer

2 bytes, read-only

This attribute provides the upstream connection of the traffic scheduler to another scheduler.

This is required when a tree of SP and WRR scheduler is being used and fed into a T-CONT scheduler.

Policy

1 byte, read, write

This attribute provides the selected scheduler policy.

The policy is configured by using the right `hw` value in case of `tc-mqprio` or automatically implied by the `tc` qdisc used.

Priority or Weight

1 byte, read, write

This is the only attribute modifiable by the OLT.

- Output
  - `priority_weight` (get)

This defines the priority or weight when a hierarchy of schedulers is being used.
In case of strict priority handling, the highest priority is 0, the lowest priority is 255.

Actions

Get, set

Notifications

This managed entity does not generate notifications.

4.20.2.12.1 Traffic Scheduler Implementation

Figure 81 presents two possible multi stage scheduling pipelines. The **T-CONT** scheduler with strict priority scheduling policy is at the root level. There are two additional schedulers in the next scheduling stage: one of the schedulers has strict priority and the other WRR policy. The third stage is only present in the model on the left, where eight more WRR schedulers are attached to the WRR scheduler in the first stage. Each of the leaf nodes has eight priority queues.

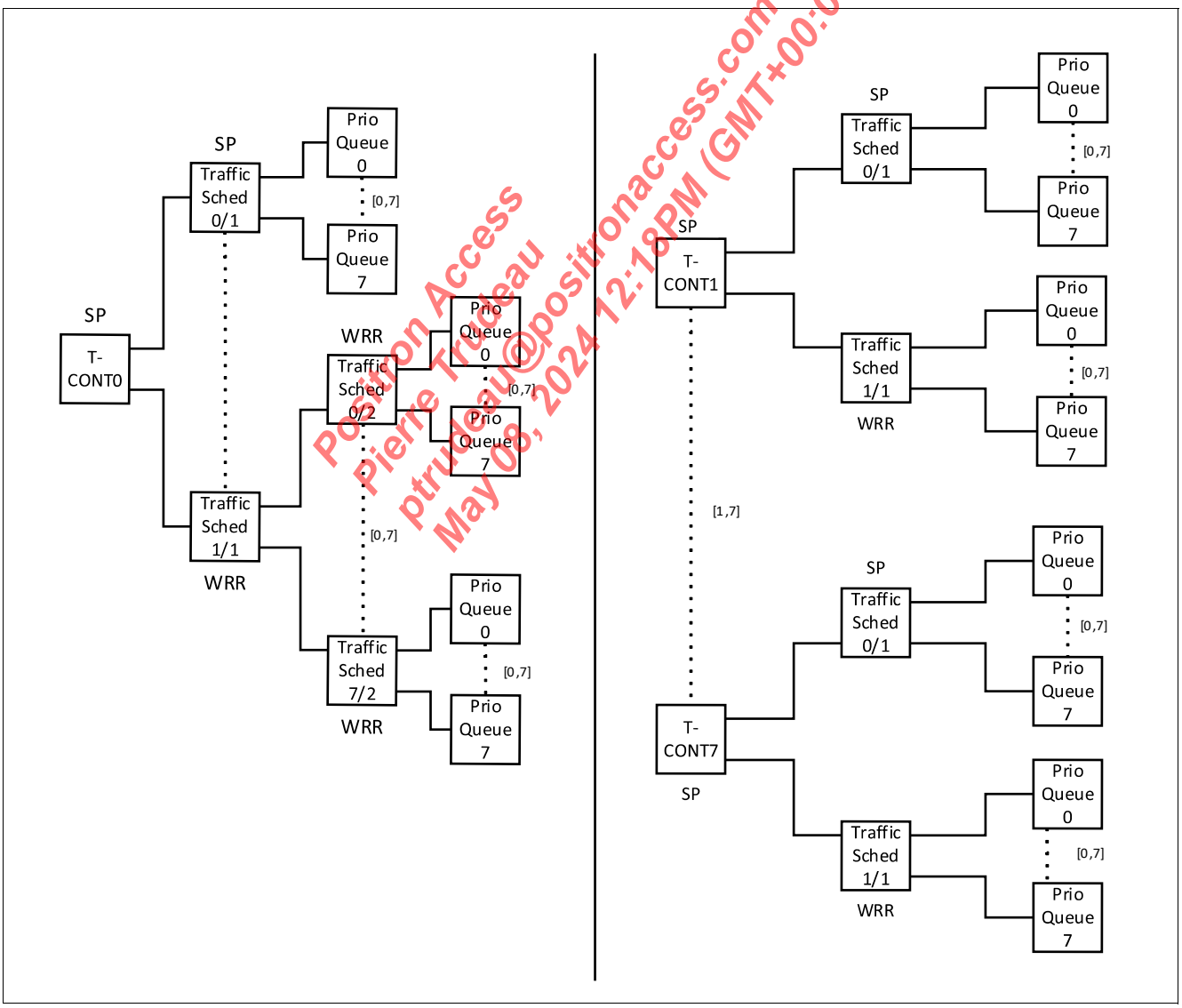


Figure 81 OMCI Multi Stage QoS Pipelines

The configuration of the multi stage QoS scheduling is done by using Linux `tc` qdiscs. **Figure 81** shows the pipeline mapped to Linux qdiscs on **Figure 82**. The root scheduler for the T-CONT is mapped to `tc-prio` qdisc. The `tc-prio` is used for the second level strict priority (SP) scheduler. The configuration of the WRR second level scheduler is done by using `tc-drr` qdisc. The second level WRR scheduler has eight WRR schedulers as children which are third level schedulers and again are configured using `tc-drr`. The leaf nodes always contain two `tc-red` qdiscs used to configure the priority queues WRED. The first is used to set the green packets thresholds and the second for the yellow packets thresholds. See **Section 4.20.2.11 Priority Queue** for details.

The node numbers in **Figure 82** are the handles of the corresponding qdiscs assigned during their creations and specific for this example. The qdisc trees are created on the T-CONT network devices for the upstream traffic.

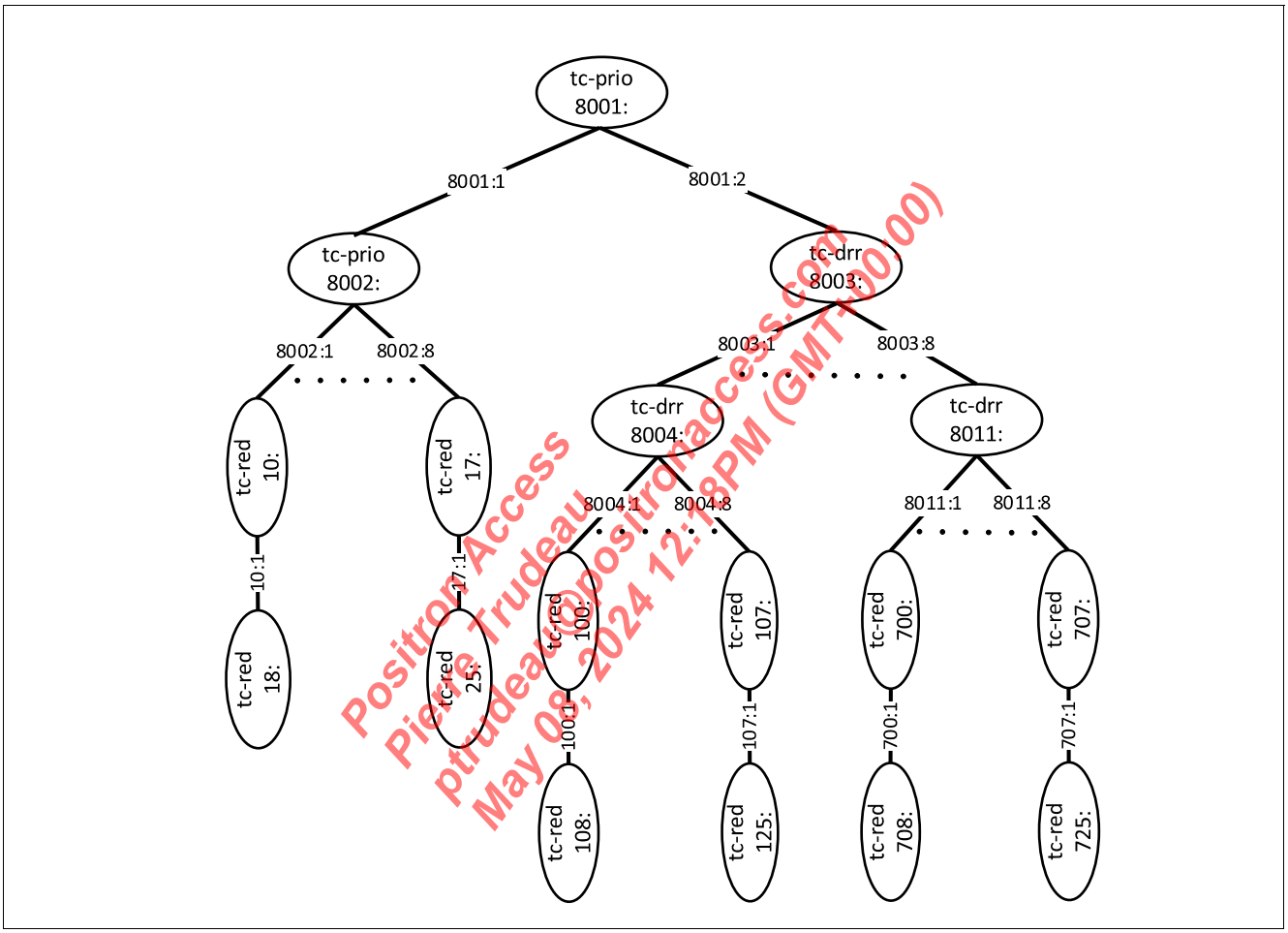


Figure 82 OMCI Multi Stage QoS Mapping to Linux qdiscs

WRR and tc-drr

The `tc-drr` requires a quantum parameter for each child class to be specified. Each class is considered as queue or output of a child scheduler. The OMCI configures a weight for each queue/scheduler. The mapping of these weights to the `tc-drr` happens by using the quanta parameter exported by the QoS driver. These values are specific to the packer processor transmit manager, constant during operation and defined in bytes. The mapping to the `tc-drr` is done by the formula:

quantum = weight * quanta

For example, when the quanta is 4 K and there are two queues with weights 1 and 2, the quantum in `tc-drr` for the first queue is 1 * 4 K and for the second 2 * 4 K. **Figure 83** shows this setup.

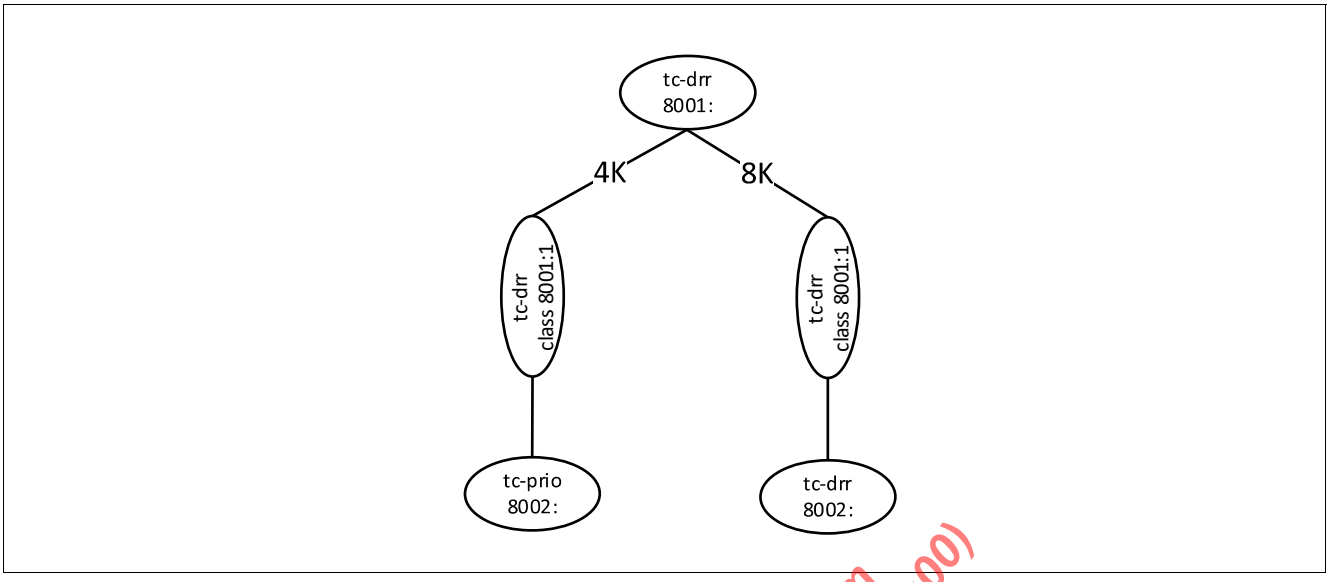


Figure 83 Linux tc-drr Configuration of Two Classes/Queues

4.20.2.13 Traffic Descriptor

ME class: 280

A traffic descriptor is created by the OLT and provides rate-based traffic management. This managed entity provides the rate definition parameters for one of the traffic descriptors. Each of the traffic descriptors can be configured individually.

The underlying hardware of the traffic descriptors depends on the context in which the traffic descriptor is used.

When used to characterize an ingress (inbound) or egress (outbound) `traffic policer`, the related hardware modules are token bucket meters. As there are two different rate values defined by this managed entity, each traffic descriptor uses two token bucket meter hardware units.

When a traffic descriptor describes an egress *traffic shaper*, the related hardware modules are token bucket shapers. In this case, two shaper entities are required to serve individual settings for the peak and committed rate and burst sizes.

Resource management is required to check for each newly created traffic descriptor that a sufficient number of token bucket meter or token bucket shaper blocks is available. If not, a negative response is given.

Note: The naming convention in the subsequent text as well as in the software driver code is meter for a policer, as there are dual token bucket meters used to implement the policing function.

Table 45 Traffic Descriptor Options

	GEM Port		Dot1 Rate Limiter		Bridge Port			
	U/S	D/S	U/S	D/S	U/S	D/S	U/S	D/S
					ANI Side		UNI Side	
Policer	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Shaper	Yes	No	No	No	No	No	No	Yes

Policer and Shaper Operation Modes

Each pair of token bucket policers or shapers is operated in one of two modes, selected by the meter type attribute.

The first mode is defined according to RFC 4115 and defines committed and excess rates as well as committed and excess bucket sizes. In this case, the excess definition is an add-on to the committed traffic and does not include it. The traffic within the committed rate/bucket size is forwarded as green. Additional traffic within the excess rate/bucket size is forwarded as yellow. The traffic beyond is discarded as red.

The second mode is defined by RFC 2698 and defines committed and a peak rates as well as maximum bucket sizes. In this case, the committed rate/bucket size is included in the peak rate/bucket size. The traffic exceeding the committed rate/bucket size but within the peak definition is forwarded as yellow. The traffic in excess of the peak definition is dropped (marked red).

Note: RFC 2698 operation mode is not supported by the PRX device family.

Managed Entity ID

2 bytes, read-only
This is an arbitrary number, selected and assigned by the OLT.

CIR

4 bytes, read, write, set-by-create
Committed information rate, in byte/s.

This value is set through the `rate` parameter of the `tc-tbf` function.

If set to 0, no rate limit is applied.

PIR

4 bytes, read, write, set-by-create
Peak information rate, in byte/s.

This value is set through the `peakrate` parameter of the `tc-tbf` function.

If set to 0, no rate limit is applied.

CBS

4 bytes, read, write, set-by-create
Committed buffer size, in byte.

This value is set through the `burst` parameter of the `tc-tbf` function.

If set to 0, a default value is applied (see `DEFAULT_COMMITTED_BURST` in `pon_net_lib`).

PBS

4 bytes, read, write, set-by-create
Peak buffer size, in byte.

This value is set through the `minburst` parameter of the `tc-tbf` function.

If set to 0, a default value is applied (see `DEFAULT_PEAK_BURST` in `pon_net_lib`).

Color Mode

1 byte, read, write, set-by-create
This attribute specifies whether the color marking algorithm considers preexisting marking on ingress packets (color-aware) or ignores it (color-blind). In color-aware mode, the packets can only be demoted (from green to yellow or red, or from yellow to red). The default value is 0 (color-blind).

- 0: Color-blind mode
- 1: Color-aware mode

Ingress Color Marking

1 byte, read, write, set-by-create
This attribute is meaningful in color-aware mode. It identifies how preexisting drop precedence is marked on ingress packets. For the DEI and PCP marking, a drop eligible indicator is equivalent to yellow; otherwise, the color is green. For the DSCP AF marking, the lowest drop precedence is equivalent to green otherwise, the color is yellow. The default value is 0.

- 0: No marking, ingress color marking is ignored
- 1: Not defined
- 2: DEI marking only
- 3: PCP 8P0D
- 4: PCP 7P1D
- 5: PCP 6D2P
- 6: PCP5P3D
- 7: DSCP AF class

Egress Color Marking

1 byte, read, write, set-by-create
This attribute specifies how drop precedence is marked by the ONU on egress packets. When set to internal marking only, the externally visible packet contents are not modified, but the packet is identified in a vendor-specific local way that indicates its color to the priority queue ME. It is possible for the egress marking to differ from

the ingress marking; for example, ingress PCP marking may be translated to DEI egress marking. The default value is 0.

- 0: No marking
- 1: Internal marking only
- 2: DEI marking only
- 3: PCP 8P0D
- 4: PCP 7P1D
- 5: PCP 6D2P
- 6: PCP5P3D
- 7: DSCP AF class

Meter Type

1 byte, read, write, set-by-create

This attribute specifies the algorithm used to determine the color of the packet. The default value is 0.

- 0: Not specified
  Select trTCM (two rate three color marker) mode according to RFC 2698.
- 1: According to IETF RFC 4115 **[46]**
  Differentiated service two rate, three color marker with efficient handling of in-profile traffic.
- 2: According to IETF RFC 2698 **[49]**
  Select trTCM (two rate three color marker) mode according to RFC 2698.
  This mode is not supported for the PRX device family.

Actions

Create, delete, get, set

Notifications

This managed entity does not generate notifications.

4.20.2.13.1 Traffic Policer Implementation

This section describes the hardware programming to configure traffic policer based on the attributes given by an OMCI traffic descriptor.

A traffic policer for downstream is configured using the QoS library function:

```
fapi_qos_classifier_set (int32_t order, cl_match_filter_t filter, cl_action_t action, uint32_t flags);
```

The policer parameters are configured using the data structure `cl_action_policer_t` in `cl_action_t`.

The traffic policers are also configured using the `tc police` qdisc **[29]**. The offloading path for the policing qdisc must be implemented. Here is an example for setting up a meter/policer on packets coming in on a UNI bridge port:

```
tc filter add dev eth0_0 ingress flower action police rate [CIR] burst [CBS] pipe police rate [PIR] burst [PBS] pipe colmark mode color-aware marker pcp_6p2d mtype trTCM
```

Note: The internal rate configuration unit is KiB/s, the smallest value is 1 KiB/s = 1024 byte/s = 8192 bit/s.

Note: The hardware supports a maximum burst size of 64 Kbytes. It is not recommended to enable the policer for upstream transmission in PON modes where the granted size is larger than 64 Kbytes, as this causes insufficient fulfillment of the upstream bursts. The usage of traffic shapers is recommended as an alternative.

4.20.2.13.2 Traffic Shaper Implementation

This section describes the hardware programming to configure a traffic shaper (rate limiter) based on the attributes given by an OMCI traffic descriptor.

A traffic shaper for downstream is configured using the QoS library:

```
int32_t fapi_qos_queue_set (char *ifname, qos_queue_cfg_t *q, uint32_t flags);
```

The shaper is configured using the data structures `qos_drop_cfg_t`, `qos_sched_mode_t` and `qos_shaper_t`.

The Linux token bucket filter `qdisc` [37] is used for upstream traffic shapers. The offloading to hardware of the `tbf qdisc` must be implemented. It is also possible to use the `tc-mqprio` shaper configuration to setup queue shapers.

4.20.2.13.3 Color Marking Mode Definitions

Table 46 shows the color marking modes based on the VLAN priority as defined by IEEE.

A VLAN priority value of 7 is the highest priority to be handled by the queue scheduling, while a value of 0 is the lowest priority.

Table 46 Color Marking Mode Programming – VLAN-based

PCP Code Point	Color Marking Mode Examples											
	8P0D			7P1D			6P2D			5P3D		
	Priority	DE	Color	Priority	DE	Color	Priority	DE	Color	Priority	DE	Color
0	0	0	Green	0	0	Green	0	0	Green	0	1	Yellow
1	1	0	Green	1	0	Green	1	0	Green	0	0	Green
2	2	0	Green	2	0	Green	2	1	Yellow	2	1	Yellow
3	3	0	Green	3	0	Green	2	0	Green	2	0	Green
4	4	0	Green	4	1	Yellow	4	1	Yellow	4	1	Yellow
5	5	0	Green	4	0	Green	4	0	Green	4	0	Green
6	6	0	Green	6	0	Green	6	0	Green	6	0	Green
7	7	0	Green	7	0	Green	7	0	Green	7	0	Green

4.20.2.14 GEM Port Network CTP Performance Monitoring History Data

ME class: 341

This managed entity is created at the request of the OLT and provides performance monitoring results that are related to a single **GEM Port Network CTP**. It replaces the deprecated **GEM Port Performance Monitoring History Data** managed entity.

The counter values are provided by the function `fapi_pon_gem_port_counters_get`.

GEM/XGEM idle frames are not included in the counter values.

Managed Entity ID

2 bytes, read-only, set-by-create

Through an identical ID, this managed entity is implicitly linked to an instance of the **GEM Port Network CTP**.

The Port ID is used to define the GEM/XGEM port for which the performance data must be delivered. The reference between GEM/XGEM port ID and GEM/XGEM Port Index is kept by the software and updated each time a **GEM Port Network CTP** is created or deleted.

Transmitted GEM Frames

4 bytes, read-only

The number of transmitted GEM/XGEM frames or frame fragments on the GEM/XGEM port, use `pa_gem_port_net_ctp_pmhd_ops` (`tx_gem_frames`).

Received GEM Frames

4 bytes, read-only

The number of transmitted GEM/XGEM frames or frame fragments on the GEM/XGEM port, use `pa_gem_port_net_ctp_pmhd_ops` (`rx_gem_frames`).

Received Payload Bytes

8 bytes, read-only

The number of transmitted GEM/XGEM bytes on the GEM/XGEM port, use `pa_gem_port_net_ctp_pmhd_ops` (`rx_payload_bytes`).

Transmitted Payload Bytes

8 bytes, read-only

The number of transmitted GEM/XGEM bytes on the GEM/XGEM port,

use `pa_gem_port_net_ctp_pmhd_ops` (`tx_payload_bytes`).

Encryption Key Errors

4 bytes, read-only

The number of XGEM encryption key errors on the XGEM port. This counter is provided for XG(S)-PON and NG-PON2 technologies only and not applicable to G-PON systems, use `pa_gem_port_net_ctp_pmhd_ops` (`key_errors`).

Key errors are not reported for G-PON (ITU-T G.984) operation mode.

Actions

Create, delete, get, get current data, set

Threshold Crossing Alert

The following Threshold Crossing Alert is generated:

TCA #1: Encryption Key Errors

4.20.3 Layer 2 Data Services

These managed entities are defined by ITU-T G.988 **[56]**, in chapter 9.3.

There are several managed entities related to the configuration of a MAC bridge and the connected bridge ports. The MAC bridges are set up in multiple ways, depending on the target application. This section provides examples of how the bridging structure looks like and points to the internal configuration that must be controlled by the OMCI software.

Section 4.20.3.1 to **Section 4.20.3.9** describe the detailed handling of the different bridge-related managed entities:

- **MAC Bridge Service Profile**
- **MAC Bridge Configuration Data**
- **MAC Bridge Port Configuration Data**
- **MAC Bridge Port Designation Data**
- **MAC Bridge Port Filter Table Data**
- **MAC Bridge Port Filter Preassign Data**
- **MAC Bridge Port Bridge Table Data**

All these managed entities are created under OLT control.

4.20.3.1 MAC Bridge Service Profile

ME class: 45

This managed entity is created by the OLT and provides attributes that control the spanning tree (**STP**) or rapid spanning tree (**RSTP**) implementation, as well as some hardware related parameters.

Note: The STP/RSTP functionality is part of the application software and therefore beyond the scope of the OMCI description.

To find out which of the bridge ports are attached to the bridge configured by this managed entity, check all **MAC Bridge Port Configuration Data** managed entities. Each of these provides a pointer to its associated bridge.

To enable quick access to this information, a list must be maintained in software and updated each time a **MAC Bridge Port Configuration Data** managed entity is added, removed, or changed.

A MAC bridge service profile must always be created before the associated **MAC Bridge Port Configuration Data** is assigned. A MAC bridge port cannot be created while the bridge does not yet exist as this results in a pointer to a non-existing managed entity, which is not allowed.

The parameters defined by the MAC bridge service profile must be stored and are used to configure the bridge each time a MAC bridge port managed entity is created or modified. This applies to the parameters:

- Learning indication
- Port bridging indication
- Unknown MAC address discard
- MAC learning depth

Managed Entity ID

2 bytes, read-only, set-by-create

The value is assigned by the OLT.

Spanning Tree Indication

1 byte, read, write, set-by-create

The boolean value true specifies that a spanning tree algorithm is enabled. The value false disables (rapid) spanning tree. The implementation is not covered by the reference software.

Learning Indication

1 byte, read, write, set-by-create

This attribute defines whether a MAC address learning must be performed by the bridge.

When learning is enabled, the number of entries to be learned can be limited. Follow the instructions given in **MAC Learning Depth**.

The learning of the bridge is disabled by setting the aging time to 0.

Port Bridging Indication

1 byte, read, write, set-by-create

This attribute defines whether bridging is allowed between UNI ports (also known as local switching).

Priority

2 bytes, read, write, set-by-create

The implementation is not covered by the reference software.

Max Age

2 bytes, read, write, set-by-create

The implementation is not covered by the reference software.

Hello Time

2 bytes, read, write, set-by-create

The implementation is not covered by the reference software.

Forward Delay

2 bytes, read, write, set-by-create

The implementation is not covered by the reference software.

Unknown MAC Address Discard

1 byte, read, write, set-by-create

This attribute defines the handling of frames having an unknown MAC destination address.

When flooding is enabled for unknown MAC destination addresses in the bridge, individual control is further provided for each of the bridge ports connected to the bridge. See **Section 4.20.3.4 MAC Bridge Port Configuration Data** for details.

MAC Learning Depth

1 byte, read, write, set-by-create

The learning limitation per bridge is configured by this attribute.

Instead of limiting the bridge learning, the learning is also limited per bridge port. This is configured by the OLT through the **MAC Bridge Port Configuration Data** managed entity.

The Linux API for configuring the MAC learning depth must be provided and added to the `iproute2` tools.

Dynamic Filtering Aging Time

4 bytes, read, write, set-by-create

This attribute configures the aging time of a bridge.

When a value of 0 is received from the OLT, a default value must be used (typically 30 s).

The default aging time is defined in the device tree file `prx300.dtsi` as:

```
intel,gsw-aging-timeout = <30>; /* seconds */
```

The aging time is applied to the hardware using the `iproute2` tools. The `switchdev` framework triggers setting the aging time configuration to the hardware when it is set on a Linux network bridge device.

For example:

```
ip link set dev sw0 type bridge ageing_time 100
```

Note: Due to a hardware limitation, only a single aging time value is configured. The latest value configured to any bridge is used by all bridge instances.

Actions

Create, delete, get, set

Notifications

This managed entity does not generate notifications.

4.20.3.1.1 MAC Bridge Setup in Linux

The OMCI stack must map the OMCI bridge ID to the Linux bridge network device name and manage this mapping because this is required to configure the bridge ports.

The bridge is initialized by using the Linux command:

```
ip link add name sw0 type bridge
```

A Linux bridge is also created using the `libnl` routing library:

```
rtnl_link_bridge_add(sk, "sw0");
```

The bridge parameters must be configurable using the bridge command. Refer to the bridge command man page for details.

Figure 84 shows a mapping of a simple OMCI bridge model to Linux. The OMCI bridge maps to the Linux software bridge and all OMCI switch ports are mapped to the Linux network devices. The figure shows also some parts of the OMCI QoS, which are **T-CONT**s (traffic containers) and queues mapping to **GEM Port Network CTP** (GEM ports). The **IEEE 802.1p Mapper Service Profile** is not represented in Linux as a network device but its VLAN priority mappings must be configured to the network bridge port device. The T-CONTs and queue configuration are mapped to a Linux network device when necessary.

Table 47 OMCI Data Path ME IDs Mappings

OMCI	PON IP Firmware	Linux/UGW DP/QoS
GEM/XGEM port ID	gem_port_id	N/A
GEM Port ME ID	gem_port_index	gem<n> (netdev)[1)2)3)]
Allocation ID	alloc_id	N/A
T-CONT ME ID	alloc_index	tcont<n> (netdev + tc qdiscs)
	qos_index	(T-CONT index & 0xff)
Bridge ME ID	N/A	sw<n> (netdev)
Bridge Port ME ID	N/A	N/A
Port number + Bridge ME ID	N/A	Included in gem/p-mapper netdev
802.1p Mapper ME ID	N/A	p-mapper<n> (netdev)
VLAN filter ME ID	N/A	tc flower filter
extVLAN ops ME ID	N/A	tc flower filter

Table 47 OMCI Data Path ME IDs Mappings (cont'd)

OMCI	PON IP Firmware	Linux/UGW DP/QoS
Priority Queue ME ID	N/A	tc qdiscs
Traffic Descriptor ME ID	N/A	tc qdisc tbf/tc filter police
VLANs	N/A	eth<n>.<vid> (netdev)

1) The creation of network devices is optional and depends mainly on the implementation requirements.
2) "n" = ME ID. For example "Bridge ME ID = 0" maps to sw0, "T-CONT ME ID = 0x8001" maps to qos_index (dequeue port) = 0x1
3) "vid" = VLAN ID

Table 47 shows the mapping of the OMCI MEs responsible to the data path configuration and how they map to configuration parameters or Linux network devices, `tc` qdiscs and filters. For using these parameters or network devices, the necessary hardware setup is done. The table does not show information about complex VLAN scenarios described in a separate chapter. The `qos_index` is specific to the PON IP and defines the egress port from which packets are dequeued when the time slot allocated for a given T-CONT comes. It is also referred as dequeue port and egress port.

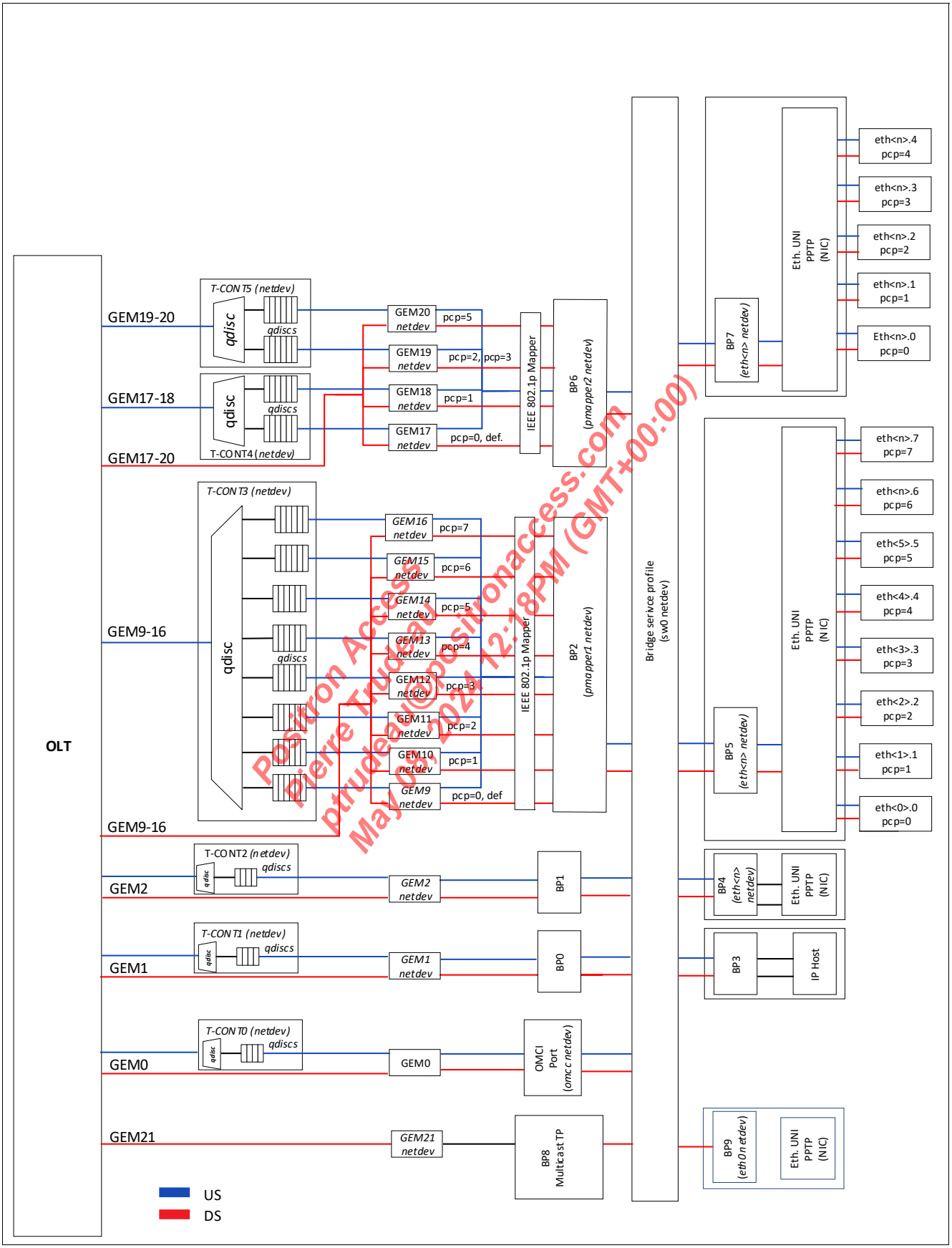


Figure 84 OMCI Bridge Mapping to Linux

The OMCI stack modules involved in the setup of the bridge network device are depicted in **Figure 85**. The configuration is done in `pon_net_lib` where a Linux bridge is created by using `libnl` **[28]**.

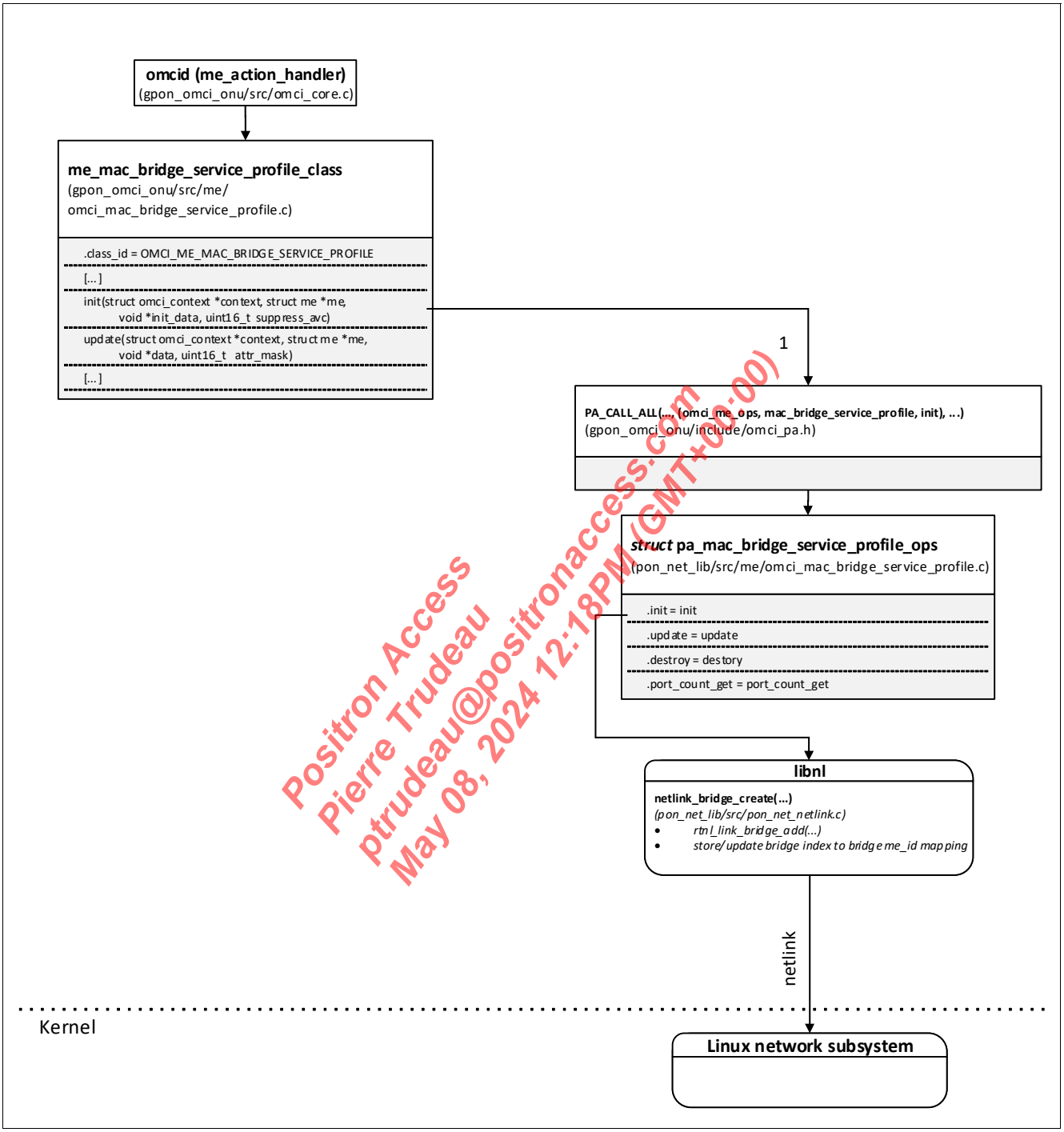


Figure 85 Modules in MAC Bridge Service Profile OMCI Configuration

4.20.3.2 MAC Bridge Configuration Data

ME class: 46

This managed entity is created by the OLT and provides information used to handle the **STP** or **RSTP** protocol.

Note: STP/RSTP functionality is part of the application software and therefore beyond the scope of this document.

Managed Entity ID

2 bytes, read-only, set-by-create

The value is assigned by the OLT. The first byte is the slot ID, in an integrated ONU this value is 0.

The second byte is the bridge group ID.

Bridge MAC Address

6 bytes, read-only

The value must be retrieved from a local definition (from config file, for example) and stored as a static entry in the MAC table to forward packets to this address to the software.

The implementation is not part of the reference software.

Bridge Priority

2 bytes, read-only

The implementation is not covered by the reference software.

Designated Root

8 bytes, read-only

The implementation is not covered by the reference software.

Root Path Cost

4 bytes, read-only

The implementation is not covered by the reference software.

Bridge Port Count

1 byte, read-only

The number of ports associated with a bridge is evaluated by scanning all valid bridge ports pointing to this managed entity.

Root Port Number

2 bytes, read-only

The implementation is not covered by the reference software.

Hello Time

2 bytes, read-only

The implementation is not covered by the reference software.

Forward Delay

2 bytes, read-only

The implementation is not covered by the reference software.

Actions

Get

Notifications

This managed entity does not generate notifications.

4.20.3.3 MAC Bridge Performance Monitoring History Data

ME class: 51

This managed entity reports the number of times that learning could not be completed due to learning limitation or MAC table overload.

A template implementation is available for reference, the functional implementation must be added by the system vendor.

Managed Entity ID

2 bytes, read-only, set-by-create

The value is assigned by the OLT. Through an identical value this managed entity is implicitly linked to a **MAC Bridge Service Profile** managed entity.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** managed entity that is pointed to by this value provides a single alarm threshold which is used for the Bridge learning entry discard count. The **Threshold Data 2** managed entity is not used.

Bridge Learning Entry Discard Count

4 bytes, read-only, set-by-create

This attribute is not implemented.

Actions

Create, delete, get, get current data, set

Notifications

The following Threshold Crossing Alert is generated:

TCA #0: Bridge learning entry discard.

4.20.3.4 MAC Bridge Port Configuration Data

ME class: 47

This managed entity is created by the OLT and provides configuration parameters related to a single bridge port.

Managed Entity ID

2 bytes, read-only, set-by-create

The value is assigned by the OLT.

Bridge ID Pointer

2 bytes, read-only, set-by-create

This attribute links the bridge port to a bridge. It points to the managed entity ID of a **MAC Bridge Configuration Data** and **MAC Bridge Service Profile**, from which the internal `bridge_index` must be retrieved.

Bridge Port Number

1 byte, read, write, set-by-create

The port number attribute assigns a number to each bridge port to identify it on the OMCI level. This must be mapped to the internal numbering through a reference table.

Termination Point Type

1 byte, read, write, set-by-create

Depending on the given termination point type, the software detects whether the bridge port is located on the UNI (LAN) or ANI (WAN) side of the bridge:

1. Physical path termination point Ethernet UNI = UNI
2. IEEE 802.1p mapper service profile = ANI
3. IP host configuration data = UNI
4. GEM interworking termination point = ANI
5. Multicast GEM interworking termination point = ANI
6. Ethernet flow termination point = UNI
7. Virtual Ethernet Interface Point (VEIP) = UNI
8. PPTP MoCA UNI = UNI
9. EFM Bonding Group

Figure 86 shows the simplified OMCI bridge model mapping to Linux bridge and network devices. The omci stack must map the **MAC Bridge Service Profile** ID to the Linux bridge device name (`sw0` in **Figure 86**). The bridge port network devices names must also be mapped according to the their number attribute ID or indexes.

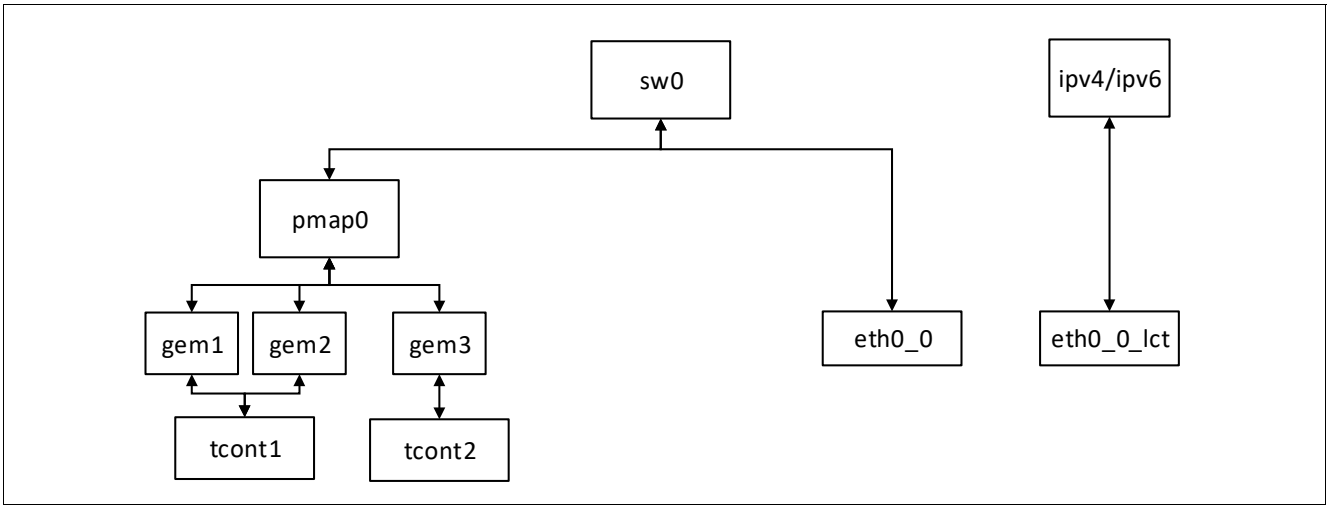


Figure 86 OMCI Bridge and Bridge Ports Mapping to Linux

For example, pmap0 may correspond to **IEEE 802.1p Mapper Service Profile** number 0 connected as termination point to the bridge `sw0`. The switch driver must support the switchdev framework to allow hardware packet offloading for the devices added to the Linux bridge network device. In the figure, the GEM nodes are connected to the ANI switch ports. They aggregate both **GEM Interworking Termination Point** and **GEM Port Network CTP**. The figure contains these network devices on the ANI side:

Two T-CONTs for T-CONT index 1 and 2, three GEM for GEM ports 1-3, one IEEE 802.1p mapper, one bridge. On the UNI side, it contains two network devices `eth0_0` and `eth0_0_lct`. The `eth0_0 device` is used for bridging of unicast traffic and `eth0_0_lct` for the **PPTP LCT UNI**. The numbers in the PON device naming may differ as the names are set during configuration according to the ME IDs.

Physical Path Termination Point Ethernet UNI Configuration in Linux

The LAN data path driver usually creates several network devices for the LAN switch port. There are dedicated devices for each kind of traffic unicast (`eth0_0`), multicast (`eth0_0_mcc`), and broadcast (`eth0_0_bcc`). There is also a special network device for the **PPTP LCT UNI** (`eth0_0_lct`).
This is an example on how the unicast network device is added to a Linux bridge using the `ip` command **[26]**. There are also kernel C APIs for that, implemented in `libnl` **[28]**. The kernel `rtnetlink` interface can be directly used to configure this.

```
ip link set eth0_0 master sw0
```

Here, `eth0_0` is the Linux network device for the UNI interface created by the LAN Ethernet data path driver.

GEM Interworking Termination Point Configuration in Linux

The bridge port network devices for the GEM ITP are special and created dynamically by OMCI. Therefore, they must be created dynamically using the netlink interface provided by the PON Ethernet data path driver. This is done using the `iproute2` tools or the kernel netlink interface. The creation of GEM network device with the `ip` command is hereafter mentioned. The parameters for the PON network device creation depend on the usage of **IEEE 802.1p Mapper Service Profile**. A p-mapper network device must be created for the **IEEE 802.1p Mapper Service Profile** and the GEM port to PCP mapping must be configured. In this case, `pcpadd` must be used in the link addition, otherwise, for untagged GEM ITPs `utadd`. Each p-mapper network device may contain up to eight GEM ITPs. One Linux network device representing the bridge port is associated with up to eight GEM/XGEM ports.
Some of the possible configurations are:

- Creating a GEM network device and linking it to tcont netdev
  ```
  ip link add link pon0 gem123 type gem id 123
  ip link set gem123 type gem id 123
  ```
- Creating a p-mapper netdevice and associate it with the GEM/XGEM ports according to pcp
  ```
  ip link add link pon0 pmapper1 type pmapper pcpdef 0
  ip link set pmapper1 type pmapper pcp 2 gem 123
  ```
- Add p-mapper to the Linux bridge
  ```
  ip link set dev pmapper1 master br0
  ```

The PON driver may provide different netlink interfaces similar to `nl80211` for the PON network device configuration. The previous APIs are provided and implemented by the PON data path driver.

The hardware offloading is enabled by adding the bridge port network device with the GEM/XGEM ports to the corresponding Linux bridge created by **MAC Bridge Service Profile**.

The previous commands are replaceable with a code using the `libnl` or netlink Linux APIs.

Here is a code example:

```
struct rtnl *br, *pon;
struct nl_cache *link_cache;
rtnl_link_alloc_cache(sk, AF_UNSPEC, &link_cache);
br = rtnl_link_get_by_name(link_cache, "sw0");
pon = rtnl_link_get_by_name(link_cache, "pmapper00");
rtnl_link_enslave(sk, br0, pon);
```

The `nl_sock` and `rtnl_link` must be allocated before calling the functions.

The hardware offloading of the data traffic happens automatically due to the switchdev support.

Attention: A p-mapper cannot be configured when there is not at least one GEM/XGEM port configured for it.

Figure 87 shows an example configuration sequence of an OMCI bridge port. The sequence corresponds to the configuration done with the Linux commands. Depending on the OMCI stack implementation, the functions are called separately and in a different order. It is also possible to do the bridge port creation and assignment in one function.

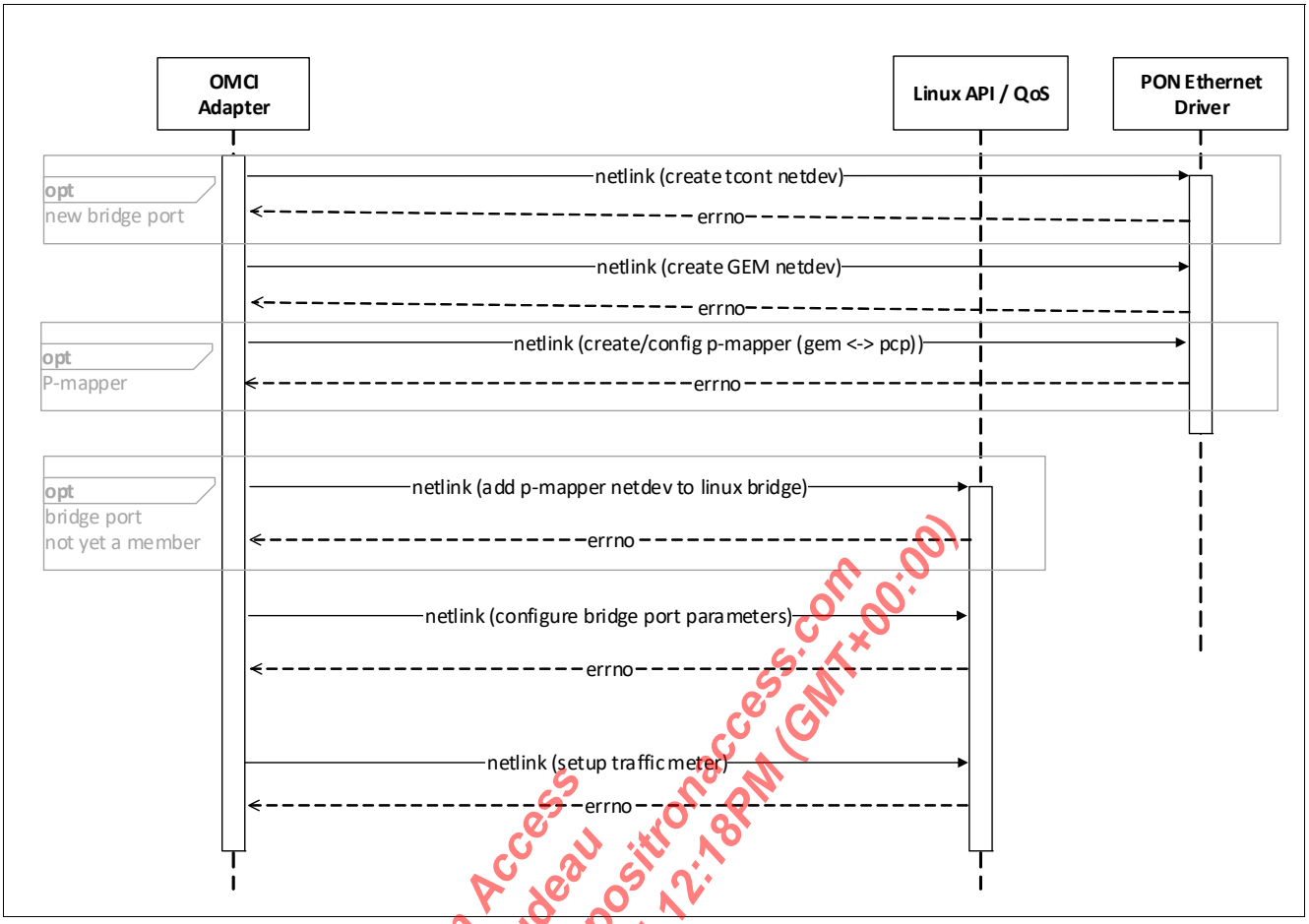


Figure 87 Example OMCI Bridge Port Configuration Sequence

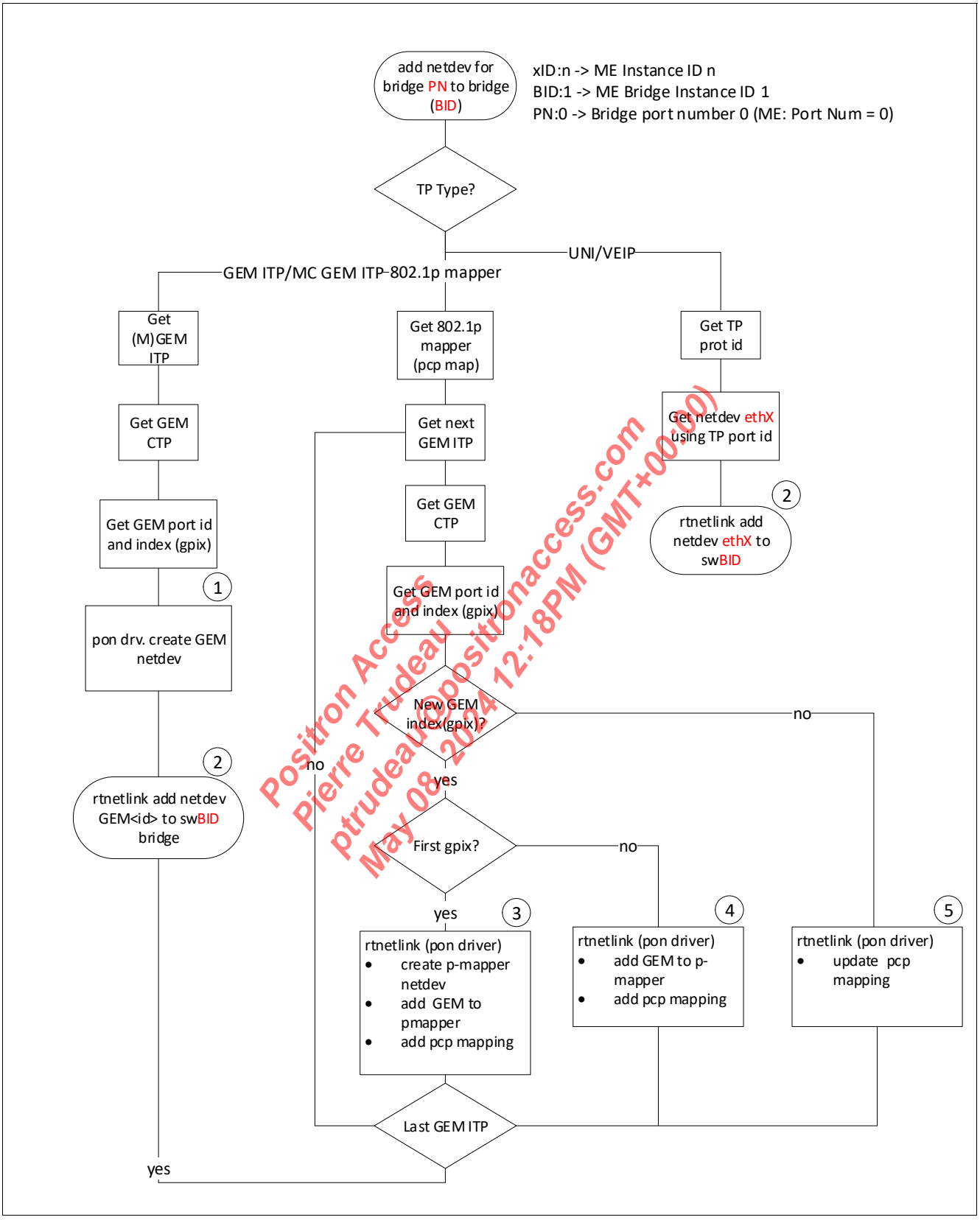


Figure 88 Bridge Port Configuration Data Setup Flow

Figure 88 shows the configuration flow for a bridge port depending on the different termination point types.

In case of **GEM Interworking Termination Point** and **Multicast GEM Interworking Termination Point**, the omci stack must get the **GEM Port Network CTP** to which the ITP MEs are pointing. Afterwards, the stack must use the GEM/XGEM port ID and index to create the GEM network device (step 1). The create network device is then added to the Linux software bridge created during the **MAC Bridge Service Profile** configuration (step 2).

In case of **PPTP Ethernet UNI** and **Virtual Ethernet Interface Point**, the network device created by the corresponding LAN or VEIP Ethernet driver must be added to the bridge (step 2). The most complex scenario is when the bridge port has **IEEE 802.1p Mapper Service Profile** attached to it. In this case, the OMCI stack must get the p-mapper instance, loop over all the PCP to GEM ITP mappings and get the respective **GEM Port Network CTP** instance. Afterwards, there are three possible configuration steps.

- Step 3 is the first GEM index mapping for the bridge port. In that case, the p-mapper and GEM network device must be created together with attaching the GEM index to the p-mapper and setting up the PCP mapping in the bridge hardware.
- Step 4 shows when the p-mapper network device is already there. Then, the GEM network device must be created, the GEM added to the p-mapper and the PCP updated.
- Step 5 shows the case when the p-mapper maps several priorities to one GEM port. In this case, only the PCP mapping for the bridge port must be updated.

Termination Point Pointer

2 bytes, read, write, set-by-create

The handling of this pointer depends on the selected **Termination Point Type**.

Port Priority

2 bytes, read, write, set-by-create

This attribute provides information purely used to handle the **STP** or **RSTP** protocol.

No hardware configuration results from this parameter.

Port Path Cost

2 bytes, read, write, set-by-create

This attribute provides information purely used to handle the **STP** or **RSTP** protocol.

No hardware configuration results from this parameter.

Port Spanning Tree Indication

1 byte, read, write, set-by-create

This attribute provides information purely used to handle the **STP** or **RSTP** protocol.

No hardware configuration results from this parameter.

Deprecated 1

1 byte, read, write, set-by-create

Deprecated 2

1 byte, read, write, set-by-create

Port MAC Address

6 bytes, read-only

This attribute reports the MAC address of the bridge port associated termination point.

For other termination point types, no MAC address is defined. A value of 0 must be reported.

Outbound Traffic Descriptor

2 bytes, read, write

This attribute defines the traffic descriptor used to limit the egress data traffic on this bridge port.

The outbound traffic policing is configured using the `tc police` qdisc. See **Traffic Descriptor** for details.

Inbound Traffic Descriptor

2 bytes, read, write

This attribute defines the traffic descriptor used to limit the ingress data traffic on this bridge port.

The outbound traffic policing is configured using the `tc police` qdisc. See **Traffic Descriptor** for details.

MAC Learning Depth

1 byte, read, write, set-by-create

This attribute is used to impose a learning limit `per bridge port`. The OMCI also defines a learning limit `per bridge` (see **MAC Learning Depth** on page **314**). The OLT must ensure no conflicting settings are applied.

LASP ID Pointer

2 bytes, read, write, set-by-create

This attribute points to an instance of the LASP managed entity.

This attribute is not implemented.

Actions

Create, delete, get, set

Alarm Notifications

This alarm is defined for this managed entity:

- Port blocking
  - This alarm is set when the STP/RSTP protocol handler decides to block a port to avoid traffic loops. The implementation is application-specific and not covered by the OMCI software package.

Other Programming

Additional configuration is available per bridge port to control the handling of data packets received on the bridge port and contain an unknown MAC destination address. When flooding is globally enabled in the related **MAC Bridge Service Profile**, it may be disabled for selected bridge ports. Flooding may be disabled independently for unknown unicast and multicast packets.

By default, these configuration bit must be set to 0.

Linux APIs

In case the STP protocol must be supported, the corresponding Linux APIs are used for that.

```
# bridge -s -d link set dev sw0p0 priority 16
# bridge -s -d link show dev sw0p0
3: sw0p0 state UP: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 16 cost 2
hairpin off guard off root_block off fastleave off learning on flood on mcast_flood
on
```

Figure 89 shows the configuration done by the OMCI stack. The **PON Adapter** operations for the bridge port configuration are implemented in `pon_net_lib`. The two operations update/connect ensure the network device corresponding to the bridge port termination point is added to the Linux bridge created by the **MAC Bridge Service Profile**. The OMCI stack adds the IEEE 802.1p mapper network devices with at least one GEM/XGEM port to the bridge. Adding a IEEE 802.1p mapper (pmapper) network device without linked GEM/XGEM port may cause problems. The calling of the connect callback is optional, the update callback is sufficient to setup the bridge port.

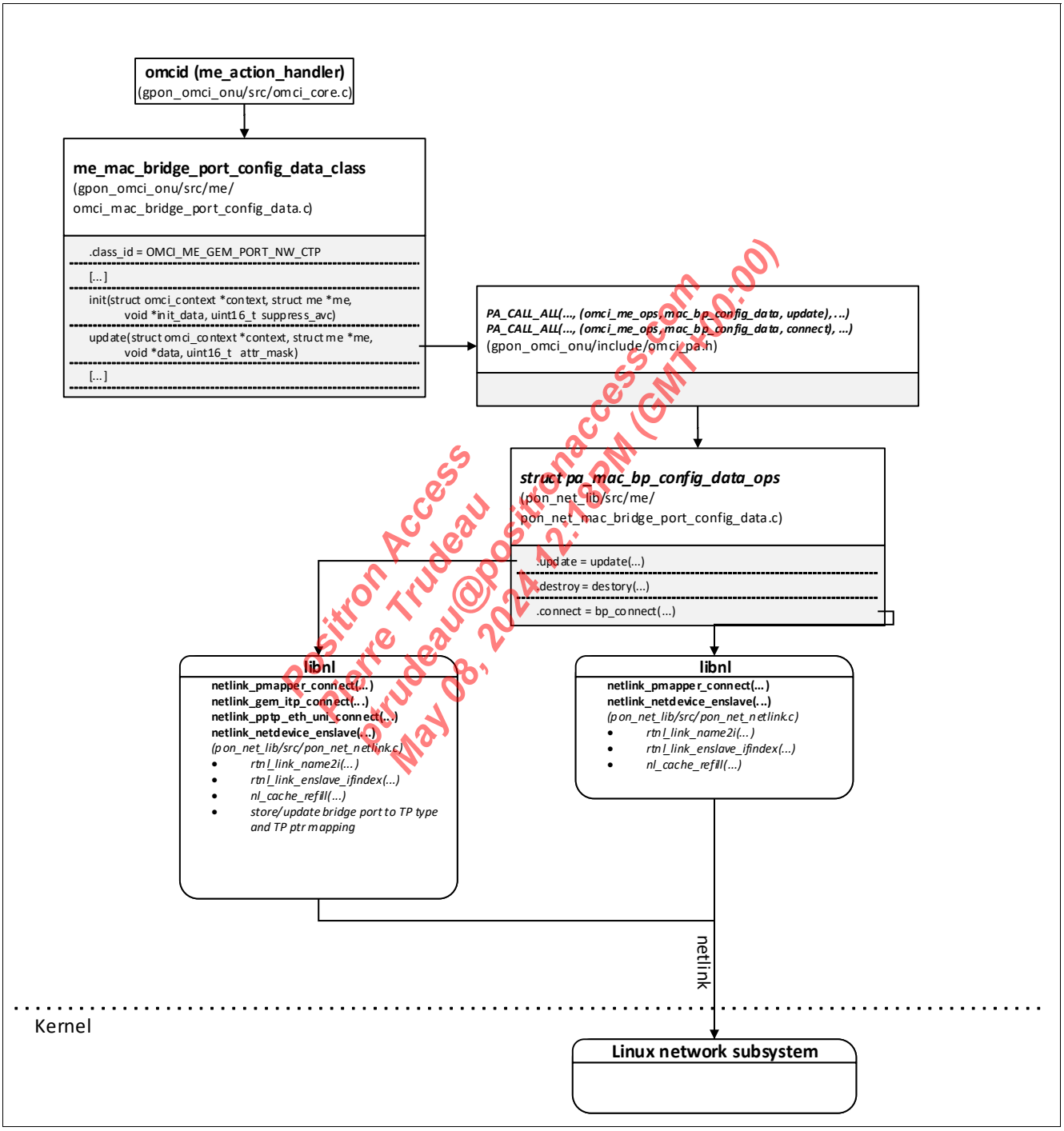


Figure 89 Modules Used in MAC Bridge Port Configuration Data Configuration

4.20.3.5 MAC Bridge Port Designation Data

ME class: 48

This managed entity is created by the OLT and provides information purely used to handle the **STP** or **RSTP** protocol. No hardware configuration results from these attributes.

Note: The STP/RSTP functionality is part of the application software and therefore beyond the scope of this document. This software may modify the port state by modification of the bridge port table.

Managed Entity ID

2 bytes, read-only, set-by-create

The value is assigned by the OLT. Through an identical value this managed entity is implicitly linked to a **MAC Bridge Port Configuration Data** managed entity.

Designated Bridge Root Cost Port

24 bytes, read-only

This attribute contains the designated root, designated cost, designated bridge and designated port, which are some of the outputs of the read port parameters operation defined in clause 14.8.2.1 of IEEE 802.1D.

Port State

1 byte, read-only

This attribute provides status information on the port. Valid values include the following:

- 0: Disabled
- 1: Listening
- 2: Learning
- 3: Forwarding
- 4: Blocking
- 5: Link down
- 6: (R)STP off

Actions

Get

Notifications

This managed entity does not generate notifications.

4.20.3.6 MAC Bridge Port Filter Table Data

ME class: 49

The MAC filter table is created by the OLT and used to filter dedicated source or destination MAC addresses.

Managed Entity ID

2 bytes, read-only, set-by-create

The value is assigned by the OLT. Through an identical value this managed entity is implicitly linked to a **MAC Bridge Port Configuration Data** managed entity.

MAC Filter Table

N x 8 bytes, read, write

The filter table attribute allows source and destination address filtering, individually per bridge port. The filtering does either work in blacklist which denies forwarding or whitelist mode which allows forwarding.

Eight bytes are provided for each line of the table attribute. The attribute received is composed of:

- Byte 1, index
  This identifies an individual entry of the table and allows to modify an entry selectively.
- Byte 2, filter control byte, with:
  - Bit 7 (LSB): Select the filter mode, 0 = forward (whitelist), 1 = filter (blacklist).
    All entries must use the same filter mode, otherwise the behavior is undefined.
  - Bit 6: Select source (1) or destination (0) address used for filtering.
  - Bit 5 to 2: Reserved, ignore these bits.
  - Bit 1 to 0: Access control.
    10: Clear the entire table (make all entries invalid).
    00: Remove this entry (make the entry invalid).
    01: Add this entry (make the entry valid).
    11: Reserved, do nothing.
- Byte 3-7: MAC address

Upon creation of this managed entity, both filter modes must be initialized to allow all (00_B).

All the previous parameters are configured using the Linux `tc` filter classifier API, either using the `iproute2` **[26]** or directly the `tc` netlink APIs. When possible, the Linux bridge command interface is used too. This is an example that allows one source MAC address and blocks all the rest:

```
$ tc filter add dev sw0p0 ingress pref 1 protocol all \
        flower indev sw0p0 \
        src_mac 3a:35:f0:ac:de:ad \
        action drop
$ tc filter add dev sw0p0 ingress pref 2 protocol all \
        flower indev sw0p0 \
        action ok
```

The `pref` parameter determines the filter execution order. This must be considered when programming the filters.

Actions

Get, get next, set, set table

Notifications

This managed entity does not generate notifications.

4.20.3.7 MAC Bridge Port Filter Preassign Data

ME class: 79

This managed entity is created by the OLT and configures filter functions for special data packet types.

By default, all filters must be inactive.

Note: The re-routing of data packets to the CPU is an application-specific option and not controlled through OMCI. The optional ingress filtering of traffic types is also not controlled by OMCI.

Managed Entity ID

2 bytes, read-only, set-by-create

The value is assigned by the OLT. Through an identical value this managed entity is implicitly linked to a **MAC Bridge Port Configuration Data** managed entity.

Common Configuration

The consequent action for a detected packet type is the same for ANI and UNI filtering, the packet is redirected to a drop queue or a drop port.

The commands to filter the different kind of packets are listed in each of the blocks listed below. The action must be changed from drop to ok to enable packet forwarding. The link layer address mask is specified in two ways described in the `iproute2` man page.

Filter IPv4 Multicast Packets

1 byte, read, write

```
$ tc filter add dev sw0p0 ingress prio 1 protocol ip\
        flower indev sw0p0 \
        dst_mac 01:00:5E:00:00:00/25 \
        action drop
```

Filter IPv6 Multicast Packets

1 byte, read, write

```
$ tc filter add dev sw0p0 ingress prio 1 protocol ipv6 \
        flower indev sw0p0 \
        dst_mac 33:33:00:00:00:00/16 \
        action drop
```

Filter IPv6 Broadcast Packets

1 byte, read, write

```
$ tc filter add dev sw0p0 ingress prio 1 protocol ipv6 \
        flower indev sw0p0 \
        dst_mac FF:FF:FF:FF:FF:FF \
        action drop
```

Filter RARP Packets

1 byte, read, write

```
$ tc filter add dev sw0p0 ingress pref 3 protocol rarp \
        flower indev sw0p0 \
        dst_mac FF:FF:FF:FF:FF:FF \
        action drop
```

Filter IPX Packets

1 byte, read, write

```
$ tc filter add dev sw0p0 ingress pref 4 protocol ipx \
        flower indev sw0p0 \
        dst_mac FF:FF:FF:FF:FF:FF \
        action drop
```

Filter NetBEUI Packets

1 byte, read, write

```
$ tc filter add dev sw0p0 ingress pref 5 protocol all \
        flower indev sw0p0 \
        dst_mac 03:00:00:00:00:01 \
        action drop
```

Filter AppleTalk Packets

1 byte, read, write

```
$ tc filter add dev sw0p0 ingress pref 6 protocol atalk \
        flower indev sw0p0 \
        dst_mac FF:FF:FF:FF:FF:FF \
        action drop

$ tc filter add dev sw0p0 ingress pref 7 protocol aarp \
        flower indev sw0p0 \
        dst_mac FF:FF:FF:FF:FF:FF \
        action drop
```

Filter Bridge Management Information Packets

1 byte, read, write

```
$ tc filter add dev sw0p0 ingress pref 8 protocol all \
        flower indev sw0p0 \
        dst_mac 01:80:C2:00:00:00/40 \
        action drop
```

Filter ARP Packets

1 byte, read, write

```
$ tc filter add dev sw0p0 ingress pref 9 protocol arp \
        flower indev sw0p0 \
        dst_mac FF:FF:FF:FF:FF:FF \
        action drop
```

Filter PPPoE Broadcast Packets

1 byte, read, write

```
$ tc filter add dev sw0p0 ingress pref 10 protocol ppp_disc \
        flower indev sw0p0 \
        dst_mac FF:FF:FF:FF:FF:FF \
        action drop
```

All of these parameters are also configured using the `tc netlink` interface APIs. The OMCI stack must take care of the filter preference so that the filter order is controlled. For hardware offloading, only the `skip_sw flag` must be passed.

Actions

Get, set

Notifications

This managed entity does not generate notifications.

4.20.3.8 MAC Bridge Port ICMPv6 Process Preassign Table

ME class: 348

This managed entity is not implemented.

4.20.3.9 MAC Bridge Port Bridge Table Data

ME class: 50

This managed entity is created by the OLT and enables the readout of the MAC forwarding table.

The complete MAC forwarding table is scanned for entries that belong to the given bridge port and the associated bridge. When matching, the MAC address, age, and dynamic indication flag are extracted from the table and reported in this managed entity.

Managed Entity ID

2 bytes, read-only, set-by-create

The value is assigned by the OLT. Through an identical value this managed entity is implicitly linked to a **MAC Bridge Port Configuration Data** managed entity.

Bridge Table

8 bytes per list entry, read-only

The contents of the MAC forwarding table is read. For an implementation reference see the code in `omci_mac_bridge_port_table_data.c`.

Actions

Get, get next

Notifications

This managed entity does not generate notifications.

4.20.3.10 MAC Bridge Port Performance Monitoring History Data

ME class: 52

This managed entity is created by the OLT and provides bridge port related performance data.

A template implementation is available for reference, the functional implementation is left open for vendor-specific solutions. See `omci_mac_bridge_port_pmhd.c`.

Managed Entity ID

2 bytes, read-only, set-by-create

The value is assigned by the OLT. Through an identical value this managed entity is implicitly linked to a **MAC Bridge Port Configuration Data** managed entity.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** managed entity that is pointed to by this value provides the alarm thresholds which are use for the Forwarded frame counter, Delay exceeded discard counter, MTU exceeded discard counter, received frame counter, and the received and discarded counter. The **Threshold Data 2** managed entity is not used.

Forwarded Frame Counter

4 bytes, read-only

This attribute is not implemented.

Delay Exceeded Discard Counter

4 bytes, read-only

No hardware counter is supported to accumulate this per bridge port.

MTU Exceeded Discard Counter

4 bytes, read-only

No hardware counter is supported to accumulate this directly per bridge port.

Received Frame Counter

4 bytes, read-only

This attribute is not implemented.

Received and Discarded Counter

4 bytes, read-only

This attribute is not implemented.

Actions

Create, delete, get, get current data, set

Notifications

The following Threshold Crossing Alerts are defined for this managed entity:

TCA #1: Delay exceeded discard

TCA #2: MTU exceeded discard

TCA #3: Received and discarded

Due to the reference implementation, these alarms are never sent.

4.20.3.11 IEEE 802.1p Mapper Service Profile

ME class: 130

This managed entity is created by the OLT and associates the priorities of VLAN-tagged frames (according to IEEE 802.1 [43]) with specific upstream GEM port connections. For untagged frames a default priority is used.

Managed Entity ID

2 bytes, read, set-by-create.

This is a unique ID, for each instance of this managed entity type, selected by the OLT.

Termination Point Pointer

2 bytes, read, set-by-create.

This attribute is not required for hardware configuration.

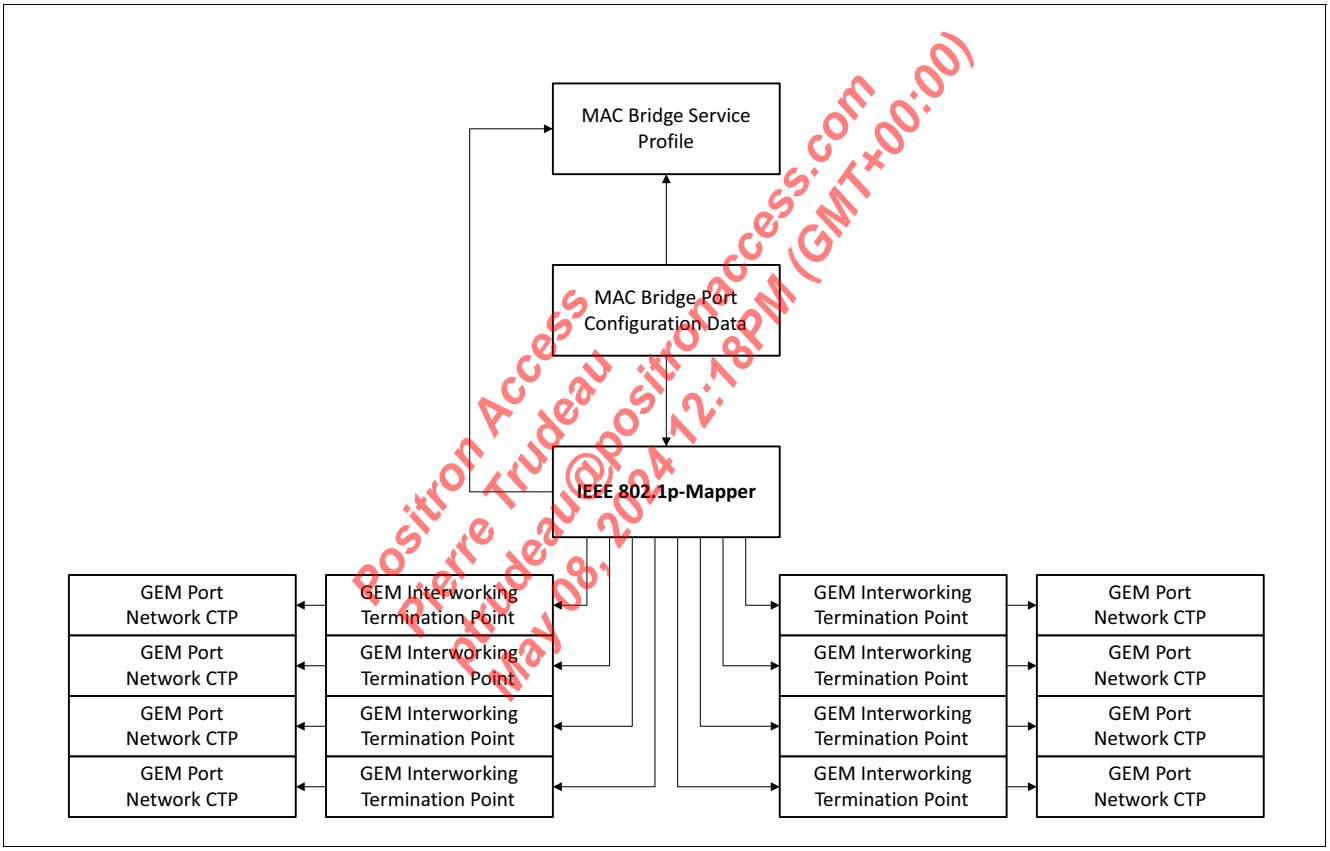


Figure 90 802.1p Mapper Managed Entity Relationship

Interworking Termination Point Pointer for P-bit Priority 0

2 bytes, read, write, set-by-create.

This attribute defines the GEM/XGEM Port used in upstream direction to carry Ethernet frames of priority code point (PCP) 0. This is the lowest priority.

Interworking Termination Point Pointer for P-bit Priority 1 to 7

2 bytes, read, write, set-by-create.

The programming sequence follows the same principle described for priority 0, with 7 used as the highest priority.

The IEEE 802.1p-mapper configuration is done by using the PON Ethernet driver p-mapper APIs.

Unmarked Frame Option

1 byte, read, write, set-by-create.

This attribute defines how the IEEE 802.1p mapper handles untagged frames.

Table 48 Unmarked Frame Option Handling

VLAN Tag	IP Packet	Unmarked Frame Option	How to select the PCP
Yes	Yes	0	Use DSCP and derive the virtual PCP from the **DSCP to P-bit Mapping** attribute.
Yes	No	0	Use the default PCP value as defined by the **Default P-bit Assumption** attribute.
No	Yes	0	Use DSCP and derive the virtual PCP from the **DSCP to P-bit Mapping** attribute.
No	No	0	Use the default PCP value as defined by the **Default P-bit Assumption** attribute.
Yes	Yes	1	Use PCP from the VLAN tag.
Yes	No	1	
No	Yes	1	Use the default PCP value as defined by the **Default P-bit Assumption** attribute.
No	No	1	

When the **Unmarked Frame Option** is set to 0, the **DSCP to P-bit Mapping** is applied if the packet carries an IP header. For non-IP packets, the **Default P-bit Assumption** is used.
When the **Unmarked Frame Option** is set to 1 and the packet carries a VLAN tag, the VLAN PCP value is used.
When the **Unmarked Frame Option** is set to 1 and the packet is untagged, the **Default P-bit Assumption** is used.

DSCP to P-bit Mapping

24 bytes, read, write.

This attribute defines the DSCP-to-P-bit mapping scheme used for untagged IP packets.

Note: The P-bit mapping is implemented at different points within the OMCI model (at the implementer choice), but must not be configured at multiple points within the same system configuration.

Default P-bit Assumption

1 byte, read, write, set-by-create.

This attribute defines the default p-bit marking assumed when the untagged frames are received by the IEEE 802.1p mapper and the `unmarked_frame_option` is set to 1. It is also used when DSCP bit mapping is selected but the frame does not contain an IP header and therefore does not carry a DSCP value field.
The packet itself is not modified.

Termination Point Type

1 byte, read, write, set-by-create.

Termination Point Type 0 – Bridged

The IEEE 802.1p-mapper GEM ITP P-bit mappings are assigned by using the `netlink` interface provided by the PON Ethernet driver. This is also done during the bridge port setup depending on the configuration sequence sent by the OLT (see **Section 4.20.3.4**).

Other termination point types are not supported.

Actions

Create, delete, get, set

Notifications

This managed entity does not generate notifications.

Implementation Hint

The implementation uses the data path library (`dp_lib`) to configure the hardware. The `dp_lib` function must only be called when at least one GEM/XGEM port has been assigned to the managed entity.
During the IEEE 802.1p mapper configuration in the OMCI stack, these configurations are done:

- Creation and configuration of the pmapper network device in the `pon_net_lib`.
- Adding of UNI network device to the bridge (optional).

Figure 91 shows the exact flow.

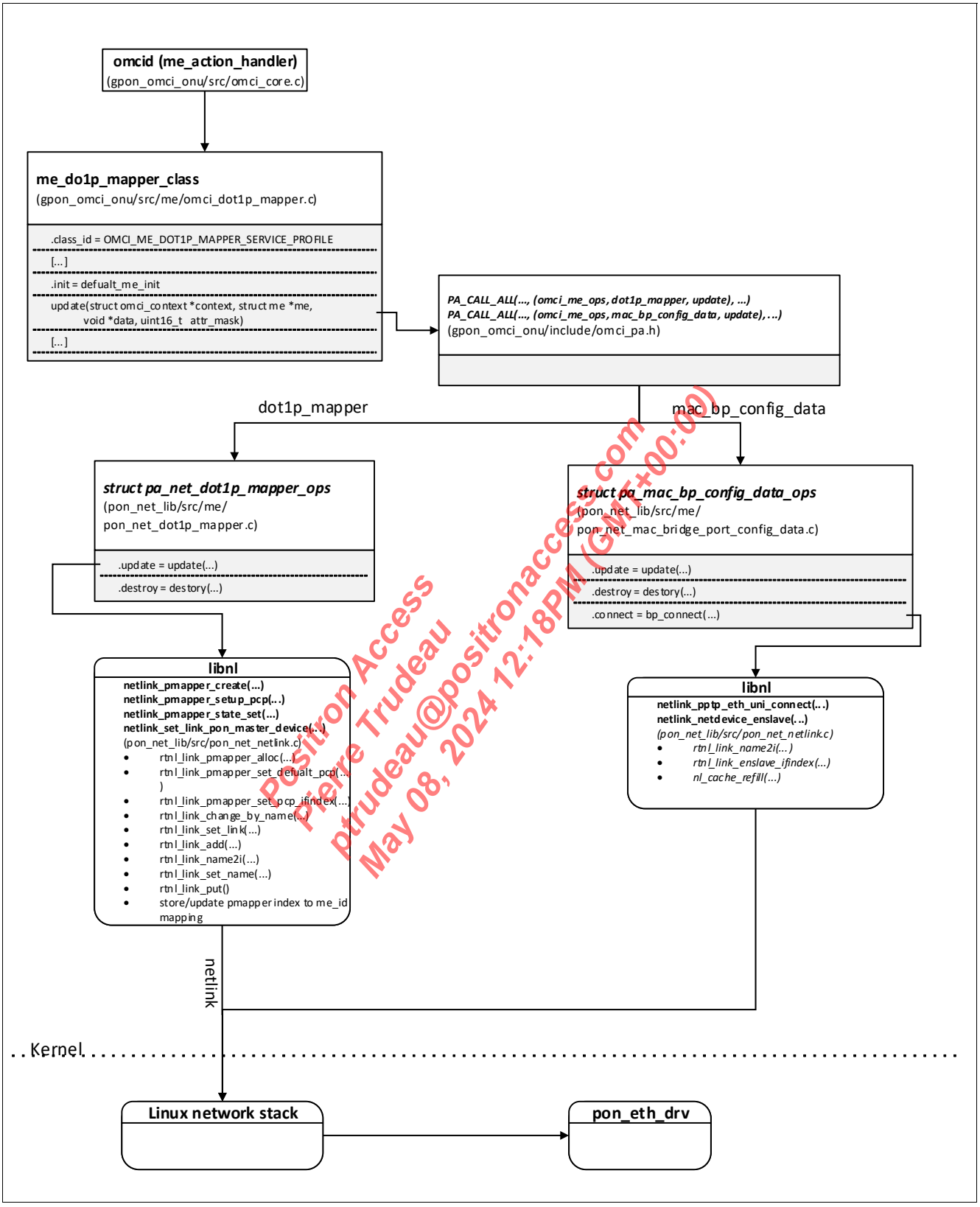


Figure 91 Modules Used in 802.1p Mapper Service Profile Configuration

Attention: Before an IEEE 802.1p mapper managed entity is removed, the related queues must be removed first.

4.20.3.12 VLAN Tagging Filter Data

ME class: 84

This managed entity is created by the OLT and defines the VLAN IDs that must be handled by the hardware. The given attributes are commonly used in upstream and downstream direction but must be configured to the ingress and egress hardware separately.

Managed Entity ID

2 bytes, read-only, set-by-create

The value is assigned by the OLT. Through an identical value this managed entity is implicitly linked to a **MAC Bridge Port Configuration Data** managed entity.

VLAN Filter List

24 bytes, read, write, set-by-create

Up to 12 two-byte values are provided by this attribute, the first N values are valid with N defined by the number of entries attribute.

Forward Operation

1 byte, read, write, set-by-create

The various actions are defined by this attribute. Depending on the selected forwarding operation, ingress and egress packet handling may vary.

Note: The use of the wording ingress and egress must be observed with special care:

The bridge port table configuration uses these classifiers from the bridge point of view; this means that packets entering the bridge are seen as ingress while packets leaving the bridge through a bridge port are seen as egress data traffic.

*In contrast, the OMCI standard uses the terms in a slightly different way: packets from UNI to the OLT (in upstream direction) are defined as ingress, while packets from the OLT to a UNI (downstream) are named egress. **Figure 92** visualizes this flow.*

*There is no difference whether a bridge port is located on the UNI side of a bridge, but the meaning is **inverted** when a bridge port is located on the ANI side of a bridge!*

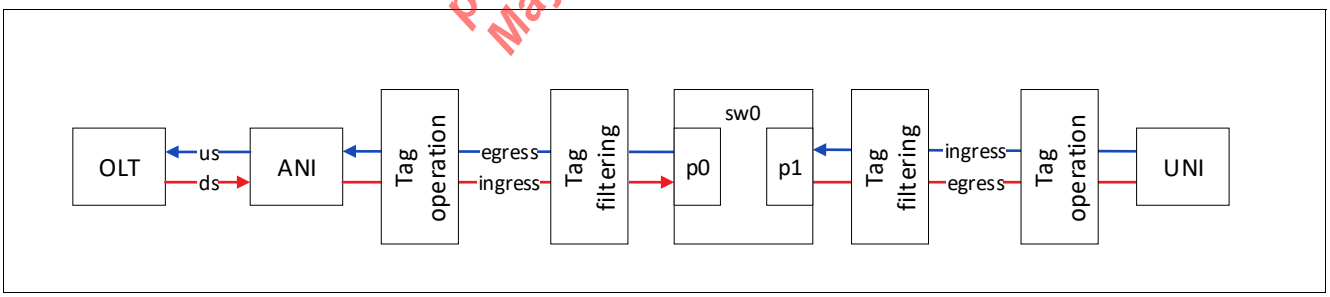


Figure 92 OMCI VLAN Filter and Tagging Operations Flow

Untagged Packets

The handling of untagged packets is configured independently for ingress and egress direction of a bridge port.

For all actions requiring untagged packet bridging, both values must be set to 0. For all actions requiring untagged packet discard, both values must be set to 1.

Number of Entries

1 byte, read, write, set-by-create

N = 1 to 12 entries are defined by this managed entity, this value defines how many are used.

The filters in **Figure 92** must be configured using `tc` and the bridge APIs of `iproute2` **[26]**. The commands are similar to the commands used in **Section 4.20.3.14**. Here are some examples for the rules:

```
# Bridging (accept all)
$ tc filter add dev sw0p0 ingress prio 1 protocol 802.1Q \
        flower indev sw0p0 skip_sw \
        action ok

# Positive filtering by VID and dropping for no match, ignore PCP
$ tc filter add dev sw0p0 ingress prio 1 protocol 802.1Q \
        flower indev sw0p0 skip_sw \
        vlan_id $1 \
        action ok
$ tc filter add dev sw0p0 ingress prio 3 protocol 802.1Q \
        flower indev sw0p0 skip_sw \
        action drop

# Negative filtering by VID, drop for match, ignore PCP
$ tc filter add dev sw0p0 ingress prio 1 protocol 802.1Q \
        flower indev sw0p0 skip_sw \
        vlan_id $1 \
        action drop
$ tc filter add dev sw0p0 ingress prio 3 protocol 802.1Q \
        flower indev sw0p0 skip_sw \
        action ok

# Positive filtering by PCP and dropping for no match, ignore VID
$ tc filter add dev sw0p0 ingress prio 1 protocol 802.1Q \
        flower indev sw0p0 skip_sw \
        vlan_prio $1 \
        action ok
$ tc filter add dev sw0p0 ingress prio 3 protocol 802.1Q \
        flower indev sw0p0 skip_sw \
        action drop

# Negative filtering by PCP, drop for match, ignore VID
$ tc filter add dev sw0p0 ingress prio 1 protocol 802.1Q \
        flower indev sw0p0 skip_sw \
        vlan_prio $1 \
        action drop
$ tc filter add dev sw0p0 ingress prio 3 protocol 802.1Q \
        flower indev sw0p0 skip_sw \
        action ok

# Positive filtering by VID & PCP and dropping for no match
$ tc filter add dev sw0p0 ingress prio 1 protocol 802.1Q \
        flower indev sw0p0 skip_sw \
        vlan_id $1 \
        vlan_prio $2 \
        action drop
$ tc filter add dev sw0p0 ingress prio 3 protocol 802.1Q \
```

```
        flower indev sw0p0 skip_sw \
        action ok

# Negative filtering by VID & PCP, drop for match
$ tc filter add dev sw0p0 ingress prio 1 protocol 802.1Q \
        flower indev sw0p0 skip_sw \
        vlan_id $1 \
        vlan_prio $2 \
        action drop
$ tc filter add dev sw0p0 ingress prio 3 protocol 802.1Q \
        flower indev sw0p0 skip_sw \
        action ok
```

When it must be provisioned to consider egress traffic too, the same rules apply but for the egress `tc` filters.

Figure 93 shows the modules involved in the VLAN tagging filter configuration. The `pon_net_lib` **PON Adapter** lower layer module implements the callbacks called by `me_vlan_tagging_filter_data_class` update method using the helper macro `PA_CALL_ALL`. The module creates the necessary tc flower filter rule according to the VLAN tagging filter data ME attributes.

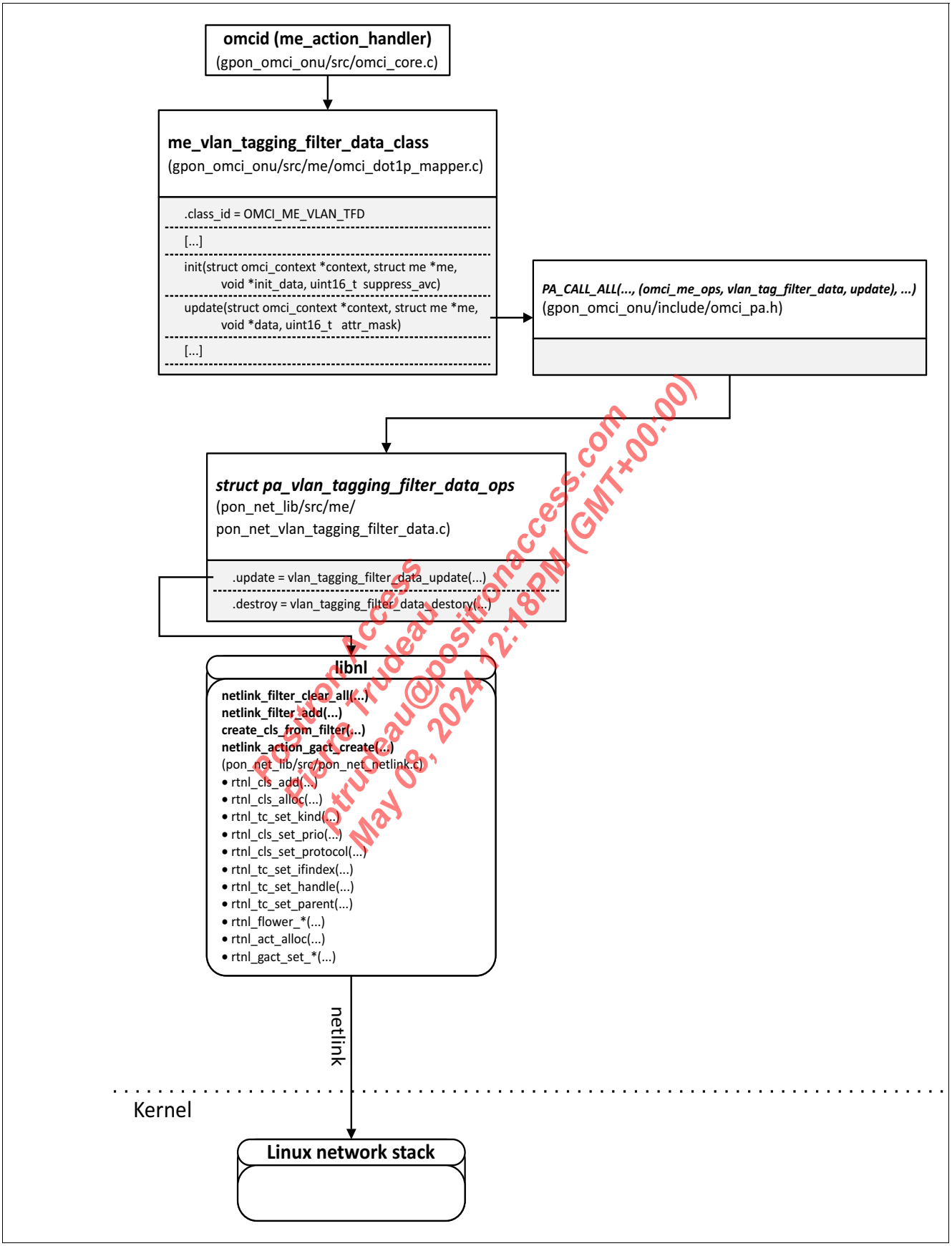


Figure 93 Modules Used in VLAN Tagging Filter Data Configuration

Actions

Create, delete, get, set

Notifications

This managed entity does not generate notifications.

4.20.3.13 VLAN Tagging Operation Configuration Data

ME class: 78

This managed entity is created by the OLT and defines the VLAN handling of incoming and outgoing frames, it can be positioned either on the UNI side of a bridge or on the ANI side and controls upstream as well as downstream actions.

In contrast to the **Extended VLAN Tagging Operation Configuration Data** managed entity this one provides less functionality in terms of VLAN manipulation and is used in older G-PON systems. These two VLAN-related managed entities are used either/or, but not concurrently in the same system application. If one of these managed entities has already been used at least once, the usage of the other one shall be answered by an error response. Both managed entity types use the same underlying hardware configuration tables.

Note: The implementation of this managed entity is not covered by the SDK.

The VLAN tagging operations can be configured by using the TC APIs of `iproute2` **[26]**. The commands are similar to the commands used in **Chapter 4.20.3.14**.

The following associations are supported, indicated by the “Association Type” attribute:

- 0: Physical Path Termination Point (PPTP) Ethernet UNI = UNI side
- 2: 802.1 p mapper Service Profile = ANI side
- 3: MAC Bridge Port Configuration Data = UNI or ANI side,
  this depends on the bridge port property stored in the Bridge Port Table
- 5: GEM Interworking Termination Point = ANI side
- 6: Multicast GEM Interworking Termination Point = ANI side
- 10: Physical Path Termination Point (PPTP) Ethernet UNI = UNI side
- 11: Virtual Ethernet Interface Point (VEIP) = UNI side
- Other code points: Not supported.

Note: Code points 0 and 10 refer to the same type of managed entity, but are handled slightly differently due to OMCI backward compatibility reasons.

Depending on the given association type, the VLAN handling definition is related to a Bridge port, to a UNI port, or to a GEM/XGEM port.

4.20.3.14 Extended VLAN Tagging Operation Configuration Data

ME class: 171

This managed entity is created by the OLT and defines the VLAN handling of incoming and outgoing frames, it is either positioned on the UNI side of a bridge/802.1p mapper, on the ANI side, or on both sides and defines the upstream actions. When the downstream operation is enabled, it must be the inverse of the upstream operation. Otherwise, the downstream data is not modified. See **Table 53** for examples.

When the data traffic is treated by this managed entity on the UNI and the ANI side, the second operation is done either on the modifications applied by the first operation (incremental operation) or the second operation is applied to the original frame (non-incremental operation).

In contrast to the deprecated VLAN tagging operation configuration data managed entity, this one provides additional functionality in terms of VLAN manipulation. These two VLAN-related managed entities are used either/or, but not concurrently in the same system application. When one of these managed entities has been used, the usage of the other one must be answered by an error response.

Note: Only this managed entity must be used for new designs. The VLAN tagging operation configuration data managed entity is not supported by the OMCI stack which is part of the 10G PON Chipset System Package.

Managed Entity ID

2 bytes, read-only, set-by-create

The value is assigned by the OLT, unique per instance of this managed entity.

Association Type

1 byte, read, write, set-by-create

These associations are supported, indicated by the association type attribute:

- 0: MAC bridge port configuration data = UNI or ANI side
- 1: 802.1 p mapper service profile = ANI side
- 2: Physical path termination point (PPTP) Ethernet UNI = UNI side
- 4: Physical path termination point (PPTP) xDSL UNI = UNI side
- 5: GEM interworking termination point = ANI side
- 6: Multicast GEM interworking termination point = ANI side
- 10: Virtual Ethernet interface point = UNI side

The internal implementation does neither provide a direct connection of an Extended VLAN ME to a bridge port nor to an 802.1p mapper ME. If this is required by the OMCI MIB, then this needs to be implemented:

- When the target bridge port is on the UNI side of the bridge, search the PPTP Ethernet UNI ME connected with the bridge port and connect the extended VLAN ME with the PPTP Ethernet UNI ME.
- When the bridge port is on the ANI side of the bridge and the bridge port is connected with a 802.1p mapper ME, the extended VLAN ME must be connected to all GEM ports associated with this 802.1p mapper.
- The same applies when the extended VLAN ME is connected directly to a 802.1p mapper ME.
- When the bridge port is on the ANI side of the bridge and the bridge port is connected directly with a GEM ITP ME, connect the extended VLAN ME with the GEM ITP ME.
- When none of the above applies, reject the setting.

The software must find out the position of the bridge port (at the ANI or UNI side of the bridge) and configure the related tables according to this information.

Received Frame VLAN Tagging Operation Table Max Size

2 bytes, read-only

This attribute indicates the maximum number of entries that can be set in the **Received Frame VLAN Tagging Operation Table** attribute.

Output TPID

2 bytes, read, write

This attribute gives the special TPID value for operations on the input (filtering) side of the table. Typical values include 0x88A8 and 0x9100.

Input TPID

2 bytes, read, write

This attribute gives the special TPID value for operations on the output (tagging) side of the table. Typical values include 0x88A8 and 0x9100.

Downstream Mode

1 byte, read, write

This selection defines, which VLAN manipulation must be applied in the downstream direction.

- **Downstream Mode 0**: Inverse operation.
  The operation performed in the downstream direction is the inverse of that performed in the upstream direction. Which treatment fields are used for downstream filtering and the handling of unmatched frames are left to the ONU implementation.
  For one-to-one VLAN mappings, the inverse is trivially defined. Many-to-one mappings are possible, however, and these are treated as follows.
  When the many-to one mapping results from multiple operation rules producing the same ANI-side tag configuration, then the first rule in the list defines the inverse operation.
  When the many-to-one mapping results from don't care fields in the filter being replaced with provisioned fields in the ANI side tags, then the inverse is defined to set the corresponding fields on the ANI side with their lowest value.
- **Downstream Mode 1**: No operation, pass all.
  Regardless of the filter rules defined for upstream, no operation is performed in the downstream direction. All downstream frames are forwarded without modification.
  – Creates three default downstream rules (for untagged, single-tagged, and double-tagged frames) that do nothing.
- **Downstream Mode 2**: Filter on VID and p-bit or pass.
  On a match, performs the inverse operation on both the VID and p-bit value.
  When no match is found, forwards the frame without modification.
  – Use the ANI-side VID and p-bit setting (from the upstream treatment) as the filter rule. Use the UNI-side VID and p-bits as treatment (from the upstream rule).
  – Add default rules to forward without change.
- **Downstream Mode 3**: Filter on VID only or pass.
  On a match, performs the inverse operation on the VID.
  When no match is found, forwards the frame without modification.
  – Proceeds as for mode 2, but ignores the p-bits in the rule definition.
- **Downstream Mode 4**: Filter on p-bit only or pass.
  On a match, performs the inverse operation on the VID.
  When no match is found, forwards the frame without modification.
  – Proceeds as for mode 2, but ignores the VID in the rule definition.
- **Downstream Mode 5**: Filter on VID and p-bit or drop.
  On a match, performs the inverse operation on both the VID and p-bit value.
  When no match is found, drops the frame.
  – Uses the ANI-side VID and p-bit setting (from the upstream treatment) as the filter rule. Uses the UNI-side VID and p-bits as treatment (from the upstream rule).
  – Adds default rules to drop the frame.

- **Downstream Mode 6**: Filter on VID only or drop.
  On a match, performs the inverse operation on the VID.
  When no match is found, drops the frame.
  – Proceeds as for mode 5, but ignores the p-bits in the rule definition.
- **Downstream Mode 7**: Filter on p-bit only or drop.
  On a match, performs the inverse operation on the VID.
  When no match is found, drops the frame.
  – Proceeds as for mode 5, but ignores the VID in the rule definition.
- **Downstream Mode 8**: Drop all.
  Regardless of the upstream filter rules, drops all downstream traffic.

*Note: Only code points 0 and 1 are defined by ITU-T G.984.4. The other code points are an enhancement done by ITU-T G.988 **[56]** and its amendments but may be used for the G.984 systems as well.*

Received Frame VLAN Tagging Operation Table

N x 16 bytes, read, write, with N being the number of table entries

This attribute is a table that filters and tags upstream frames. Each entry represents a tagging rule, comprising a filtering part (the first eight fields) and a treatment part (the last seven fields). Each incoming upstream packet is matched against each rule in list order. The first rule that matches the packet is selected as the active rule, and the packet is then treated according to that rule.

The details of the rule handling are described in **Section 4.20.3.14.1** to **Section 4.20.3.14.5**.

Associated ME Pointer

2 bytes, read, write, set-by-create

This attribute points to the managed entity with which this extended VLAN tagging operation configuration data managed entity is associated.

DSCP to P-bit Mapping

25 bytes, read, write

This attribute defines the mapping of DSCP values to IEEE 802.1p priority bit values. The attribute usage is the same as for the **IEEE 802.1p Mapper Service Profile**.

Enhanced Mode

1 byte, read-only, set-by-create

The boolean value true specifies that the enhanced received frame classification and processing table is used, and the received frame VLAN tagging operation table is ignored.

Note: The Enhanced Mode is not supported by the reference implementation.

Enhanced Received Frame Classification and Processing Table

N x 28 bytes, read, write, with N being the number of table entries

Note: The Enhanced Mode is not supported by the reference implementation.

Actions

Create, delete, get, get next, set, set table

Notifications

This managed entity does not generate notifications.

4.20.3.14.1 Rule Definition

From the OMCI viewpoint, the extended VLAN managed entity implicitly defines the behavior in ingress and egress direction, where the action in one direction is the reverse action in the other direction. For the hardware configuration, this must be configured separately. Thus, a single setting is required when the downstream function is disabled, or one setting per direction when a downstream and upstream VLAN modification is used.

Default Rules

At the time this managed entity is created, three default rules per direction (upstream and downstream) must be configured autonomously. One for untagged, one for single-tagged, and one for double-tagged packets. These rules must have the lowest possible priority and must pass the data without modification. The OLT may modify these rules at a later time. Deletion of a default rule by the OLT must be inhibited. When no default rule exists and no other rule matches, the data packet is dropped.

Rule Sorting

The OMCI definition requires the received rules to be sorted based on the first 8 bytes of the received VLAN tagging operation table. The definition of this sorting is ambiguous and the OLT expectations differ between OLT vendors. See **VLAN Rule Sorting** in **Section 3.5** for details.

4.20.3.14.2 Upstream Rule Definition

This section describes the construction of the upstream rule set, directly defined by the given attributes.

Attention: The OMCI considers the tag of a single-tagged frame to be the inner tag.

Note: A rule is deleted when all treatment attribute bits are set to 1. Padding bits are ignored and set to 1 or 0.

Input TPID – Upstream

This attribute defines a TPID value, the VLAN Ethertype, used to limit the scope of a rule to VLANs of exactly this type. Only four dedicated values are supported by the hardware, other values cannot be used:

- 0x8100 - fixed in hardware
- 0x88A8 - configured during switch initialization
- 0x9100 - configured during switch initialization
- 0x9200 - configured during switch initialization

To allow these TPID types to pass the parsing procedure and identify the subsequent bytes as VLAN tags, the TPID values are handled by the GWSIP parser firmware and the last three values are in addition configured into GSWIP during initialization. The value of 0x8100 is fixed by the GSWIP hardware. All these values cannot be changed by vendor software.

Input TPID Filter Value Modification

Note: This is an example that must be executed manually from the command line.
The switch API is available to handle this directly in software.

Change the second value - from 0x88A8:

```
switch_cli gsw_register_set nregaddr=0x44e ndata=2
switch_cli gsw_register_set nregaddr=0x44a ndata=${NEW_VAL}
switch_cli gsw_register_set nregaddr=0x44f ndata=0x8020
switch_cli gsw_register_set nregaddr=0xa42 ndata=${NEW_VAL}
```

Change the third value - from 0x9100:

```
switch_cli gsw_register_set nregaddr=0x44e ndata=3
switch_cli gsw_register_set nregaddr=0x44a ndata=${NEW_VAL}
switch_cli gsw_register_set nregaddr=0x44f ndata=0x8020
switch_cli gsw_register_set nregaddr=0xa4a ndata=${NEW_VAL}
```

Change the fourth value - from 0x9200:

```
switch_cli gsw_register_set nregaddr=0x44e ndata=4
switch_cli gsw_register_set nregaddr=0x44a ndata=${NEW_VAL}
switch_cli gsw_register_set nregaddr=0x44f ndata=0x8020
switch_cli gsw_register_set nregaddr=0xa4b ndata=${NEW_VAL}
```

Input TPID Filter Value Readout

Note: This is an example that must be executed manually from the command line.
The switch API is available to handle this directly in software.

Read the second value (the default value is 0x88A8):

```
# switch_cli gsw_register_set nregaddr=0x44e ndata=2
GSW_REGISTER_SET
# switch_cli gsw_register_set nregaddr=0x44f ndata=0x8000
GSW_REGISTER_SET
# switch_cli gsw_register_get nregaddr=0x44a
```

```
GSW_REGISTER_GET
Returned values:
----------------
                        nRegAddr:       1098 (0x044a)
                        nData:         34984 (0x88a8)
```

Read the third value - the default value is 0x9100:

```
# switch_cli gsw_register_set nregaddr=0x44e ndata=3
GSW_REGISTER_SET
# switch_cli gsw_register_set nregaddr=0x44f ndata=0x8000
GSW_REGISTER_SET
# switch_cli gsw_register_get nregaddr=0x44a
GSW_REGISTER_GET
Returned values:
----------------
                        nRegAddr:       1098 (0x044a)
                        nData:         34984 (0x9100)
```

Read the fourth value - the default value is 0x9200:

```
# switch_cli gsw_register_set nregaddr=0x44e ndata=4
GSW_REGISTER_SET
# switch_cli gsw_register_set nregaddr=0x44f ndata=0x8000
GSW_REGISTER_SET
# switch_cli gsw_register_get nregaddr=0x44a
GSW_REGISTER_GET
Returned values:
----------------
                        nRegAddr:       1098 (0x044a)
                        nData:         34984 (0x9200)
```


Output TPID – Upstream

This attribute defines a TPID value, the VLAN Ethertype, used instead of the default type 8100_H when modifying a VLAN tag in egress direction. Only the four dedicated values as listed for the input TPID are supported by the hardware, other values cannot be used.

VLAN Filter Configuration

The filter portion of a rule decides to which ingress packets the treatment portion of that rule may apply. When a packet matches the filter portion of a rule, the index of this rule is used to select the rule treatment.

When none of the rules matches and a default treatment is defined for the incoming packet type (untagged, single-tagged or double-tagged), the default treatment is applied. When no default rule exists, the data packet is dropped.

Note: OMCI recommends that by default three default rules are defined by the ONU, for untagged, single-tagged, and double-tagged packets, respectively.

The basic filter definition is defined by the attributes filter outer priority and filter inner priority. **Table 49** to **Table 51** show this as a matrix.

Table 49 VLAN Filter Configuration – Rule Overview

Filter Outer Priority	Filter Inner Priority				
	0 … 7	**8**	**9 … 13**	**14**	**15**
0 … 7	Double-tagged rule	Double-tagged rule	*Invalid rule*	Double-tagged rule	*Invalid rule*
8	Double-tagged rule	Double-tagged rule	*Invalid rule*	Double-tagged rule	*Invalid rule*
9 … 13	Invalid rule				
14	Double-tagged rule	Double-tagged rule	*Invalid rule*	**Default double-tagged rule**	*Invalid rule*
15	Single-tagged rule	Single-tagged rule	*Invalid rule*	**Default single-tagged rule**	**Default untagged rule**

The default rules are stored as the last three entries of a rule set. Each time a new default rule is received, the existing one is overwritten.

Table 50 VLAN PCP Filter Modes

Filter Outer Priority	Filter Inner Priority				
	0 … 7	**8**	**9 … 13**	**14**	**15**
0 … 7	Filter on both inner and outer PCP values, as given by the attributes. This rule applies to **double-tagged** packets.	Filter on outer PCP value only, as given by the attribute. This rule applies to **double-tagged** packets.	This is not a valid configuration.	Do not filter on the inner PCP value. This rule applies to **double-tagged** packets.	This is not a valid configuration.
8	Filter on inner PCP value only, as given by the attribute. This rule applies to **double-tagged** packets.	Do not filter on any PCP, neither inner nor outer. This rule applies to **double-tagged** packets.	This is not a valid configuration.	Do not filter on any PCP value. This rule applies to **double-tagged** packets.	This is not a valid configuration.
9 … 13	This is not a valid configuration.				
14	Do not filter on the outer PCP value. This rule applies to **double-tagged** packets.	Do not filter on any PCP value. This rule applies to **double-tagged** packets.	This is not a valid configuration.	Do not filter on any PCP, neither inner nor outer. This rule applies to **double-tagged** packets to define the **default** rule.	This is not a valid configuration.
15	Filter on inner PCP value only, as given by the attribute. This rule applies to **single-tagged** packets.	Do not filter on the PCP value. This rule applies to **single-tagged** packets.	This is not a valid configuration.	Do not filter on the PCP value. This rule applies to **single-tagged** packets to define the **default** treatment.	Filtering on PCP does not apply. This rule defines the **default** treatment for **untagged** packets.

Table 51 VLAN Filter Configuration – Implementation

Filter outer priority	Filter inner priority				
	0 … 7	**8**	**9 … 13**	**14**	**15**
0 … 7	outer_priority_enable = true outer_priority_value = 0…7 inner_priority_enable = true inner_priority_value = 0…7 zero_enable = false one_enable = false two_enable = true default = false	outer_priority_enable = true outer_priority_value = 0…7 inner_priority_enable = false inner_priority_value = 0 zero_enable = false one_enable = false two_enable = true default = false	This is not a valid configuration.	outer_priority_enable = false outer_priority_value = 0 inner_priority_enable = true inner_priority_value = 0…7 zero_enable = false one_enable = false two_enable = true default = false	This is not a valid configuration.
8	outer_priority_enable = false outer_priority_value = 0 inner_priority_enable = true inner_priority_value = 0…7 zero_enable = false one_enable = false two_enable = true default = false	outer_priority_enable = false outer_priority_value = 0 inner_priority_enable = false inner_priority_value = 0 zero_enable = false one_enable = false two_enable = true default = false	This is not a valid configuration.	outer_priority_enable = false outer_priority_value = 0 inner_priority_enable = false inner_priority_value = 0 zero_enable = false one_enable = false two_enable = true default = false	This is not a valid configuration.
9 … 13	This is not a valid configuration.				

Table 51 VLAN Filter Configuration – Implementation (cont'd)

Filter outer priority	Filter inner priority				
	0 … 7	**8**	**9 … 13**	**14**	**15**
14	outer_priority_ enable = true outer_priority_ value = 0…7 inner_priority_ enable = false inner_priority_ value = 0 zero_enable = false one_enable = false two_enable = true default = false	outer_priority_ enable = false outer_priority_ value = 0 inner_priority_ enable = false inner_priority_ value = 0 zero_enable = false one_enable = false two_enable = true default = false	This is not a valid configuration.	outer_priority_ enable = false outer_priority_ value = 0 inner_priority_ enable = false inner_priority_ value = 0 zero_enable = false one_enable = false two_enable = true **default = true**	This is not a valid configuration.
15	outer_priority_ enable = true outer_priority_ value = 0…7 inner_priority_ enable = false inner_priority_ value = 0 zero_enable = false one_enable = true two_enable = false default = false	outer_priority_ enable = false outer_priority_ value = 0 inner_priority_ enable = false inner_priority_ value = 0 zero_enable = false one_enable = true two_enable = false default = false	This is not a valid configuration.	outer_priority_ enable = false outer_priority_ value = 0 inner_priority_ enable = false inner_priority_ value = 0 zero_enable = false one_enable = true two_enable = false **default = true**	outer_priority_ enable = false outer_priority_ value = 0 inner_priority_ enable = false inner_priority_ value = 0 zero_enable = true one_enable = false two_enable = false **default = true**

Filter Outer VID – Upstream

This rule parameter defines how the VLAN ID field of an outer VLAN tag must be used to filter an incoming frame.

This setting has no effect on single-tagged or untagged frames.

Filter Outer TPID/DE – Upstream

This rule parameter defines how the VLAN TPID and DE fields of an outer VLAN tag must be used to filter an incoming frame.

This setting has no effect on single-tagged or untagged frames.

Filter Inner VID – Upstream

This rule parameter defines how the VLAN ID field of an inner VLAN tag must be used to filter an incoming frame.

This setting has no effect on untagged packets.

Filter Inner TPID/DE – Upstream

This rule parameter defines how the VLAN TPID and DE fields of an inner VLAN tag must be used to filter an incoming frame.

This setting has no effect on untagged packets.

Filter Ethertype – Upstream

This rule parameter defines how the Ethertype field must be used to filter an incoming frame for VLAN treatment.

It is possible to configure multiple different Ethertype values, although OMCI currently only defines these:

- 0: No Ethertype filtering
- 1: 0800_H (IPv4)
- 2: 8863_H or 8864_H (PPPoE)
- 3: 0806_H (ARP)
- 4: $86DD_H$ (IPv6)
- 5 to 15: Not defined

For OMCC versions that do not yet support the dedicated Ethertype configuration for IPv6 (0xA0 and earlier), any setting defined for IPv4 must also be applied to IPv6 packets arriving with Ethertype $86DD_H$.

This is achieved by adding an additional rule for the IPv6 Ethertype $86DD_H$ with the same treatment than the corresponding IPv4 Ethertype 0800_H. This special handling implementation is at the ONU vendor discretion.

Treatment Tags to Remove – Upstream

This attribute defines whether received VLAN tags must be removed.

The treatment programming depends on the filter programming, as untagged, single-tagged, and double-tagged packets must be treated differently.

It is the OLT responsibility to configure useful treatments. The ONU does not check for or reject meaningless configurations, such as removing VLAN tags from untagged packets.

Table 52 VLAN Tag Removal Treatment Programming

Tag Removal Treatment	Filter Rule Type		
	Untagged	**Single-tagged**	**Double-tagged**
0: Do not remove any VLAN	discard_enable = 0 outer_not_generate = 1 inner not generate = 1	discard_enable = 0 outer_not_generate = 0 inner not generate = 1	discard_enable = 0 outer_not_generate = 0 inner not generate = 0
1: Remove the outer VLAN tag	discard_enable = 0 outer_not_generate = 1 inner not generate = 1	discard_enable = 0 outer_not_generate = 1 inner not generate = 1	discard_enable = 0 outer_not_generate = 1 inner not generate = 0
2: Remove the two outermost VLAN tags	discard_enable = 0 outer_not_generate = 1 inner not generate = 1	discard_enable = 0 outer_not_generate = 1 inner not generate = 1	discard_enable = 0 outer_not_generate = 1 inner not generate = 1
3: Delete the data packet	discard_enable = 1 outer_not_generate = 1 inner not generate = 1	discard_enable = 1 outer_not_generate = 1 inner not generate = 1	discard_enable = 1 outer_not_generate = 1 inner not generate = 1

Treatment Outer Priority – Upstream

This attribute defines how the priority of an outer VLAN tag must be handled in egress direction.

This setting is used when two tags must be added, to define the priority of the outermost VLAN tag.

Treatment Outer VID – Upstream

This rule parameter defines how the VLAN ID of an outer VLAN tag must be handled in egress direction.

This setting is used when two tags must be added, to define the VID of the outermost VLAN tag.

Treatment Outer TPID/DE – Upstream

This rule parameter defines how the TPID and DE flag of an outer VLAN tag must be handled in egress direction.

This setting is used when two tags must be added, to define the TPID and DE of the outermost VLAN tag.

Treatment Inner Priority – Upstream

This rule parameter defines how the priority of an inner VLAN tag must be handled in egress direction.

Treatment Inner VID – Upstream

This rule parameter defines how the VLAN ID of an inner VLAN tag must be handled in egress direction.

Treatment Inner TPID/DE – Upstream

This rule parameter defines how the TPID and DE flag of an inner VLAN tag must be handled in egress direction.

Received Frame VLAN Tagging Operation Table Details

The configuration data previously described is handled as a single table attribute holding four 32-bit double-words of data. **Figure 94** shows a structural overview. The reserved bit fields are ignored. Any different use of reserved fields is up to the ONU vendor software handling but must be avoided for interoperability reasons.

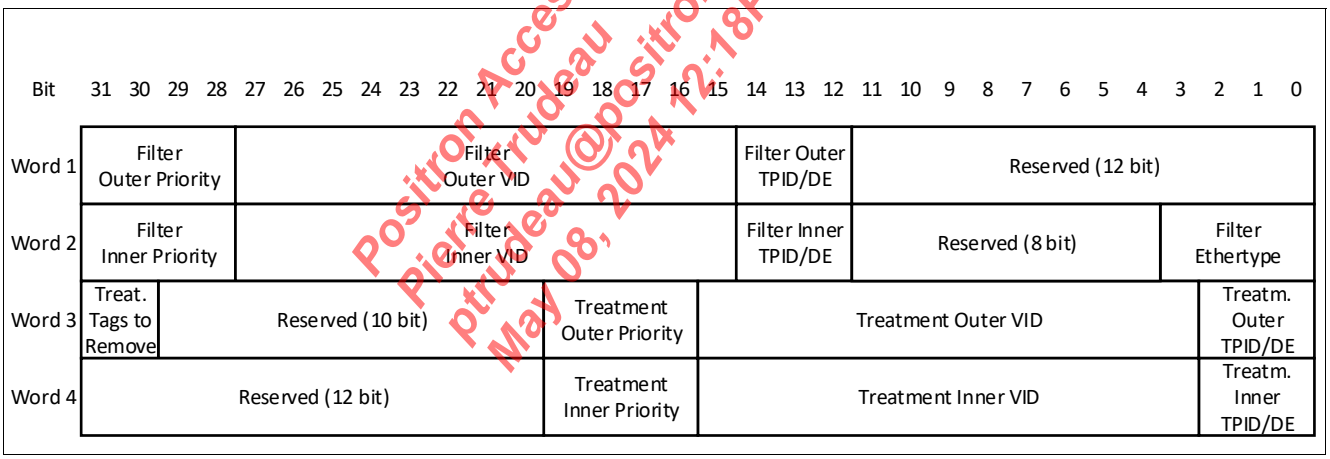


Figure 94 Received Frame VLAN Tagging Operation Table

4.20.3.14.3 Downstream Rule Definition

This section describes the construction of the downstream rule set, defined as the inverse operation described by the given attributes. The downstream handling depends on a combination of upstream attribute settings and cannot be directly derived from individual managed entity attributes. Table 53 describes, for some common configurations, how the inverse downstream rule is derived from the upstream rule.

Attention: The OMCI considers the tag of a single-tagged frame to be the inner tag.

Table 53 Rule Description

No.	Upstream	Downstream
Untagged Upstream Packets		
0.1	Do nothing (default rule for untagged packets).	When untagged, do nothing (default rule for untagged packets).
0.2	Insert one full tag X to an untagged packet.	When single-tagged with X, remove the tag.
0.3	Insert one full tag X for a given Ethertype.	When single-tagged with X, remove the tag.
0.4	Insert two full tags (outer = Y, inner = X) to all untagged packets.	When double-tagged with Y and X, remove two tags.
Single-tagged Upstream Packets		
1.1	Do nothing (default rule for single-tagged packets).	When single-tagged, do nothing (default rule for single-tagged packets).
1.2	Add one full tag X to all single-tagged packets, copy the priority.	When double-tagged with outer tag X and any priority, remove the outer tag.
1.3	Add one full tag X to all single-tagged packets when the tag is C.	When double-tagged with outer tag X and inner tag C, remove the outer tag.
1.4	Add one tag X to all single-tagged packets when the tag is C, copy priority of C to X.	When double-tagged with outer tag X and inner tag C, remove the outer tag.
1.5	Insert two full tags (outer = Y, inner = X) to all single-tagged packets.	When double-tagged with Y and X, remove two tags.
1.6	Insert two full tags (outer = Y, inner = X) to single-tagged packets when tag is C.	When double-tagged with Y and X, remove two tags.
1.7	Modify tag to X with fixed priority, when tag is C.	Modify tag to C with copied priority, when tag is X.
1.8	Modify tag to X with copied priority, when tag is C.	Modify tag to C with copied priority, when tag is X.
1.9	Modify tag to X with fixed priority, when tag is C and priority is Pc.	Modify tag to C with copied priority, when tag is X.
1.10	Modify the existing tag from C to X and add another one Y.	When double-tagged with Y and X, remove the outer tag and change the inner tag to C.
1.11	Remove the tag, when it is C.	When untagged, add a single tag C with default priority.
1.12	Remove the tag, when it is C and the priority is Pc.	When untagged, add a single tag C with priority Pc.
1.13	Remove the tag, regardless of its VLAN ID or priority.	When untagged, add a single tag C with default priority.
Double-tagged Upstream Packets		
2.1	Do nothing (default rule for double-tagged packets).	When double-tagged, do nothing (default rule for double-tagged packets).
2.2	Add another tag X with fixed priority Px, when the outer tag is S and the inner tag is C.	When double-tagged with X (priority Px) and S (any priority), remove outer tag.

Table 53 Rule Description (cont'd)

No.	Upstream	Downstream
2.3	Add another tag X with priority copied from outer tag, when the outer tag is S and the inner tag is C.	When double-tagged with X (any priority) and S, remove outer tag.
2.4	Add an outer tag Y with fixed priority Py and an inner tag X with fixed priority Px, when the outer tag is S and the inner tag is C.	When double-tagged with Y and X (priorities Py and Px), remove the two tags.
2.5	Add an outer tag Y with copied priority (Ps) and an inner tag X with copied priority (Pc), when the outer tag is S and the inner tag is C.	When double-tagged with Y and X (any priorities), remove the two tags.
2.6	Modify the outer tag to new VID X and priority Px, when tags are S and C.	When double-tagged with X and C, modify outer tag to S with default priority.
2.7	Modify the outer tag to new VID X and keep priority Sx, when tags are S and C.	When double-tagged with X and C, modify outer tag to S with copied priority from Px.
2.8	Modify both tags S and C to new values Y and X, with fixed priorities.	When double-tagged with Y and X (and priorities Py and Px), change tags to S and C with default priorities.
2.9	Modify both tags S and C to new values Y and X, keep the priorities.	When double-tagged with Y and X (and priorities Py and Px), change tags to S and C with copied priorities.
2.10	Swap both tags.	Swap both tags.
2.11	When double-tagged with S an C, remove the outer tag.	When single-tagged, add an outer tag S with default priority.
2.12	When double-tagged with S an C, remove the both tags.	When untagged, add an outer tag S and an inner tag C with default priorities.

4.20.3.14.4 OMCI Extended VLAN Configuration in Linux

The OMCI extended VLAN configuration must be done using Linux traffic control (TC) classifier action subsystem (CA). The user space tool to show and manipulate the TC control settings `tc` is part of the `iproute2` utilities. The TC works with network devices (netdev) and its classifier capabilities must be used to program the OMCI extended VLAN configurations. The TC subsystem consists of these blocks:

- Queuing disciplines (qdiscs) are scheduling objects either classful or classless. When classful, the qdisc has multiple classes which are selected by classifier filters. Given classful qdiscs are able to contain other qdiscs, it is possible to setup a hierarchy to allow differentiated treatment of packet groupings as defined by policy. Each qdisc is identified via a 32-bit classid.
- Classes are either queues or qdiscs. The qdiscs further allow for more hierarchies as illustrated. The parent qdisc (in the hierarchy) schedules its inner qdiscs/queues using some defined scheduling algorithm. Each class is identified via a 32-bit classid.
- Classifiers are selectors of packets. They inspect either packet data or meta data and select an action to execute. The classifiers are anchored on qdiscs or classes. Each classifier type implements its own algorithm and is specialized. A classifier contains filters which implement semantics applicable to the classifier algorithm. For each policy defined, there is a built-in filter which matches first based on the layer 2 protocol type.
- Actions are executed when a resulting classifier filter matches. The most primitive action is the built-in classid/flowid selector action. Its role is to sort out which class/flow a packet belongs to and where to multiplex to in the policy graph.

The TC is able to bypass the Linux kernel network stack but it is also able to put the packets back for reclassification.

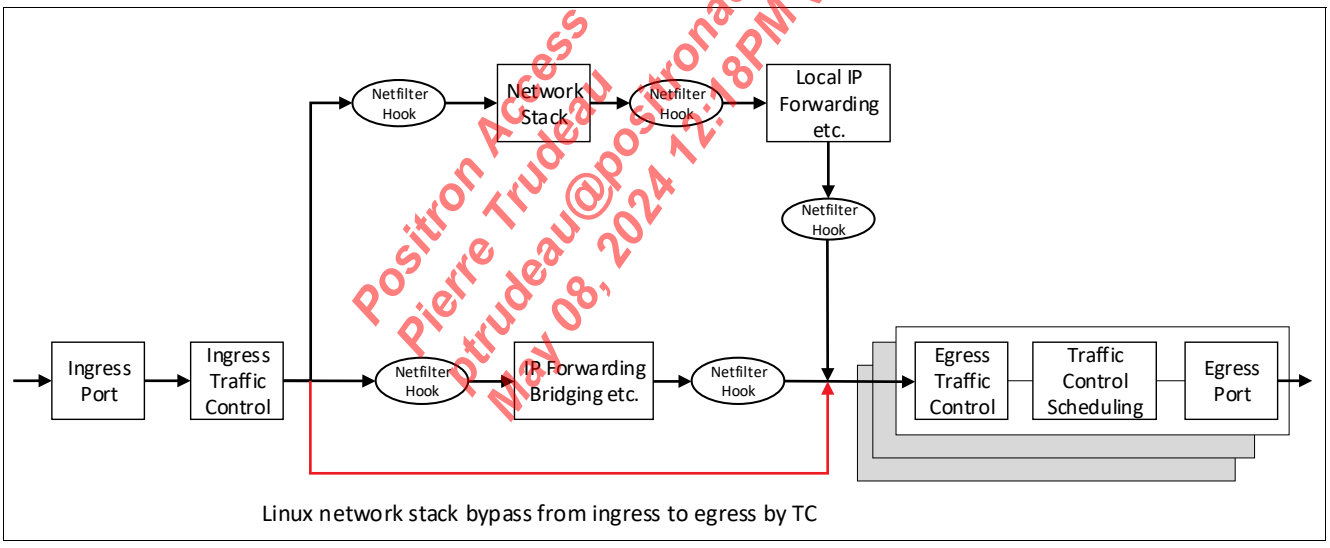


Figure 95 Traffic Control Classifier Action Data Flow

4.20.3.14.5 TC Software Implementation for OMCI

The classifying filter capabilities of the TC must be used for implementing the extended VLAN configurations. There are already several TC filter implementations, including `tc-flower`. The OMCI VLAN configurations are easily mapped using this filter together with the `tc-vlan` action. The filter also supports hardware offloading. It uses a kernel API that passes the filter and action arguments directly to the underlying hardware. It is able to do hardware offloading and software processing of the packets. All three combinations are possible: software processing, hardware offloading and both offloading and software processing. The `tc-flower` is issued as reference implementation or extended for the OMCI needs. The switch driver is also extended to support the TC hardware offloading.

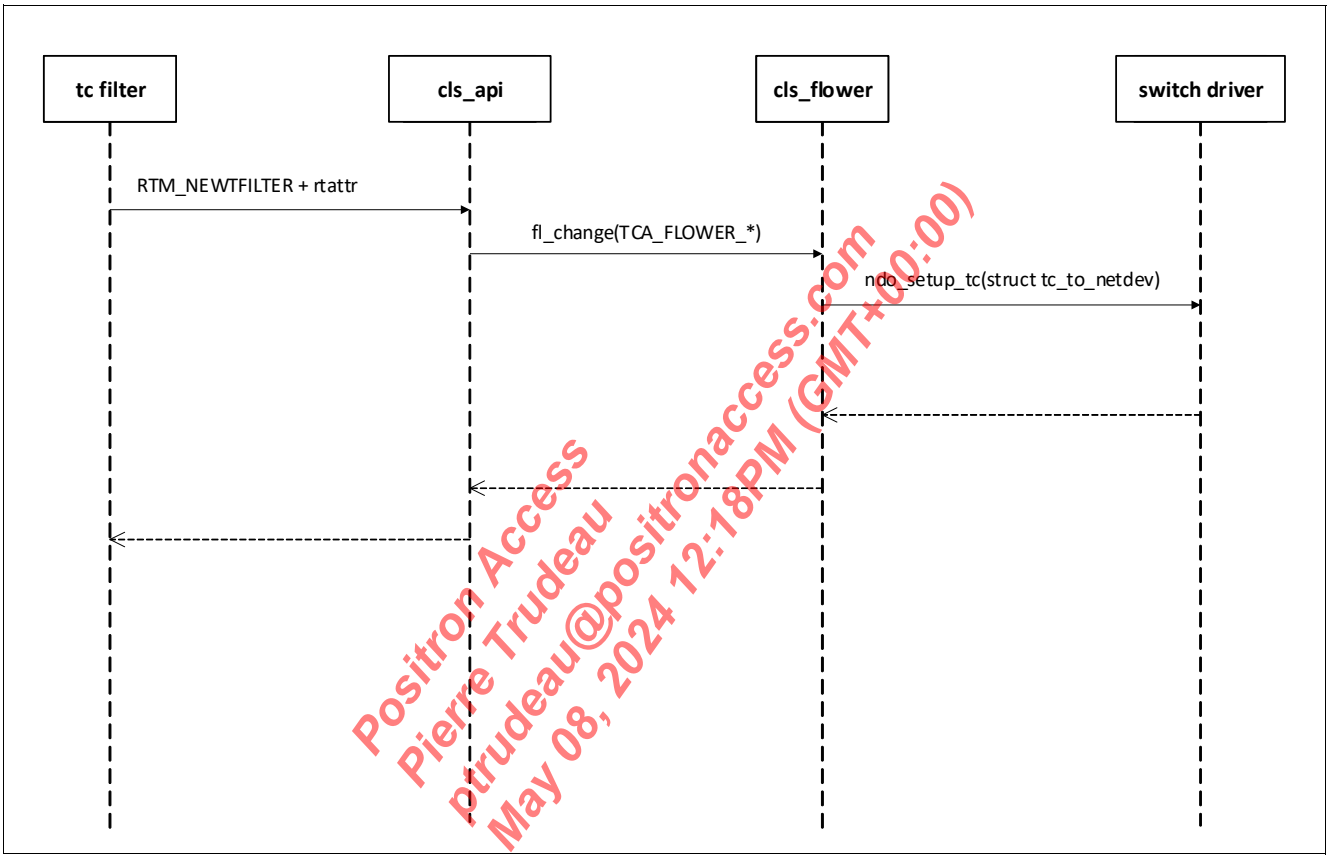


Figure 96 Flower Traffic Classifier Configuration Example with the tc Command

Switch Driver Extensions for TC

The network devices provided by the switch driver implement this `net_device_ops` callback.

```
int (*ndo_setup_tc)(struct net_device *dev, u32 handle, __be16 protocol, struct
tc_to_netdev *tc);
/* Called to setup any 'tc' scheduler, classifier or action on @dev.
 * This is always called from the stack with the rtnl lock held and netif tx queues
stopped.
 * This allows the netdevice to perform queue management safely. */
struct tc_to_netdev {
       unsigned int type;
       union {
              struct tc_cls_u32_offload *cls_u32;
              struct tc_cls_flower_offload *cls_flower;
              struct tc_cls_matchall_offload *cls_mall;
```

```
                struct tc_cls_bpf_offload *cls_bpf;
                struct tc_mqprio_qopt *mqprio;
};
        bool egress_dev;
}
```

The `ndo_setup_tc` receives all the necessary information for the VLAN configuration from the `struct tc_to_netdev`. The structure contains all the information received from the `tc` command. This information is also passed using netlink sockets in the same kernel interface as the `iproute2` utilities. The currently supported `flower` matching key attributes is found in the kernel header file `pkt_cls.h`. These attributes are extended to support multiple VLAN matching rules. At the moment, only the single VLAN matching rule is supported which is not enough to support all possible OMCI configurations. Refer to Linux man pages for details about the matching lists used by `flower`. **Figure 96** shows an example of call sequence of `flower` filter configuration. The figure shows the complete sequence from issuing the `tc` command to applying the configuration to the switch. The next sections describe the example `tc` rules for the most common OMCI configurations. The recommended way for configuration from the OMCI daemon is the `rtnetlink` API but it is also possible to call the `tc` command directly.

Extended VLAN Configurations for Untagged Frames

The OMCI extended VLAN operations for untagged frames are configured by using the `tc-flower` filter and `tc-vlan` action. These are examples for the UNI interface `$eth="eth0_0"`

- Drop all untagged ingress frames [rule 0.3 ingress].
  `tc filter add dev $eth ingress protocol all flower action drop`
- Insert one full tag X = 333 with fixed priority = 3 [rule 0.5 ingress].
  `tc filter add dev $eth ingress protocol all flower action vlan push id 333 priority 3 protocol 0x8100`
- When single-tagged with X = 333 and priority = 3, remove outer tag X (rule 0.5) [rule 0.5 egress].
  `tc filter add dev $eth egress protocol all flower vlan_id 333 vlan_prio 3 action vlan pop`
- Insert one full tag X = 888 with fixed priority = 2 for given ethtype = 0x0800 (ipv4) [rule 0.8 ingress].
  `tc filter add dev $eth ingress protocol ip flower action vlan push id 888 priority 2 protocol 802.1q`
- When single-tagged with X = 888, priority = 2 and ethtype = 0x0800 (ipv4), remove tag X [rule 0.8 egress].
  `tc filter add dev $eth egress protocol 802.1q flower vlan_id 888 vlan_prio 2 vlan_ethtype ip action vlan pop`

Note: All the commands must be called with the `tc-flower skip_sw` *flag in order to offload the rules to the hardware.*

Extended VLAN Configurations for Single-tagged Frames

The OMCI extended VLAN operations for single-tagged frames can be configured by using `tc-flower` filter and `tc-vlan` action. These are examples for the UNI interface `$eth="eth0_0"`:

- Drop all single-tagged ingress frames [rule 1.4 ingress].
  ```
  tc filter add dev $eth ingress protocol 802.1q flower action drop
  ```
- Insert one tag X = 1000 with priority = 6 to all single-tagged [rule 1.7 ingress].
  ```
  tc filter add dev $eth ingress protocol 802.1q flower action vlan push id 1000
  priority 6
  ```
- Double-tagged with outer tag X = 1000 and priority = 6, remove one tag X = 1000 [rule 1.7 egress].
  ```
  tc filter add dev $eth egress protocol 802.1q flower vlan_id 1000 vlan_prio 6
  vlan_ethtype 802.1q cvlan_id 100 action vlan pop
  ```
- Insert one full tag X = 2000 when outer tag is C = 100 ingress [rule 1.11 ingress].
  ```
  tc filter add dev $eth ingress protocol 802.1q flower vlan_id 100 action vlan push
  id 2000
  ```
- When double-tagged inner tag is C = 100, remove outer tag egress [rule 1.11 egress].
  ```
  tc filter add dev $eth egress protocol 802.1q flower vlan_ethtype 802.1q cvlan_id
  100 action vlan pop
  ```
- Modify tag when tag is C = 100 (change tag C = 200, prio = 6, TPID = 802.1ad) [rule 1.17 ingress].
  ```
  tc filter add dev $eth ingress protocol 802.1q flower vlan_id 100 action vlan
  modify id 200 priority 6 protocol 802.1ad
  ```
- When single-tagged and tag is X = 200, modify to C = 100 and copy prio [rule 1.17 egress].
  ```
  tc filter add dev $eth egress protocol 802.1ad flower vlan_id 200 action vlan
  modify id 100
  ```
- Modify tag when tag is C = 100 (new tag X = 200, keep priority) for ARP packets [rule 1.18 ingress].
  ```
  tc filter add dev $eth ingress protocol 802.1q flower vlan_id 100 vlan_ethtype arp
  action vlan modify id 200
  ```
- When single-tagged and tag X = 200, modify to tag C = 100 and copy priority for ARP packets [rule 1.18 egress].
  ```
  tc filter add dev $eth egress protocol 802.1ad flower vlan_id 200 vlan_ethtype arp
  action vlan modify id 100
  ```
- Modify tag when tag is anything with given prio (new tag X = 200, new priority = 5) rule [1.26 ingress].
  ```
  tc filter add dev $eth ingress protocol 802.1ad flower vlan_prio 1 action vlan
  modify id 200 priority 5
  ```
- When single-tagged and tag is X, modify to 0 and set prio [rule 1.26 egress].
  ```
  tc filter add dev $eth egress protocol 802.1ad flower vlan_id 200 action vlan
  modify id 0 priority 1
  ```
- Replace the current tag with X = 2000, whatever it is, based on Ethertype = ipv6 [rule 1.33 ingress].
  ```
  tc filter add dev $eth ingress protocol 802.1q flower vlan_ethtype ipv6 action
  vlan modify id 2000 priority 1
  ```
- When tagged with X = 2000 and priority = 1 and Ethertype = ipv6, replace with zero tag [rule 1.33 egress].
  ```
  tc filter add dev $eth egress protocol 802.1ad flower vlan_id 2000 vlan_prio 1
  vlan_ethtype ipv6 action vlan modify id 0 priority 0
  ```

Note: All the commands must be called with the tc-flower skip_sw flag in order to offload the rules to the hardware.

Extended VLAN Configurations for Double-tagged Frames

The OMCI extended VLAN operations for double-tagged frames are configured by using the `tc-flower` filter and `tc-vlan` action. These are examples for the UNI interface `$eth="eth0_0"`:

- Drop all double-tagged frames [rule 2.4 ingress]
  ```
  tc filter add dev $eth ingress protocol 802.1ad flower action drop
  ```
- Modify the outer tag to new X = 200 and keep prio for any inner tag and inner prio [rule 2.16 ingress]
  ```
  tc filter add dev $eth ingress protocol 802.1q flower vlan_id 100 vlan_ethtype
  802.1q action vlan modify id 200
  ```
- When double-tagged with outer tag X = 200, modify the outer tag and keep outer prio [rule 2.16 egress]
  ```
  tc filter add dev $eth egress protocol 802.1ad flower vlan_id 200 vlan_ethtype
  802.1q action vlan modify id 100
  ```

Note: All the commands must be called with the `tc-flower skip_sw` *flag in order to offload the rules to the hardware.*

DSCP to P-bit Mapping

Some of the extended VLAN rules require deriving the VLAN P-bit from the DSCP. **Figure 97** shows an example of DSCP ro P-bit mapping.

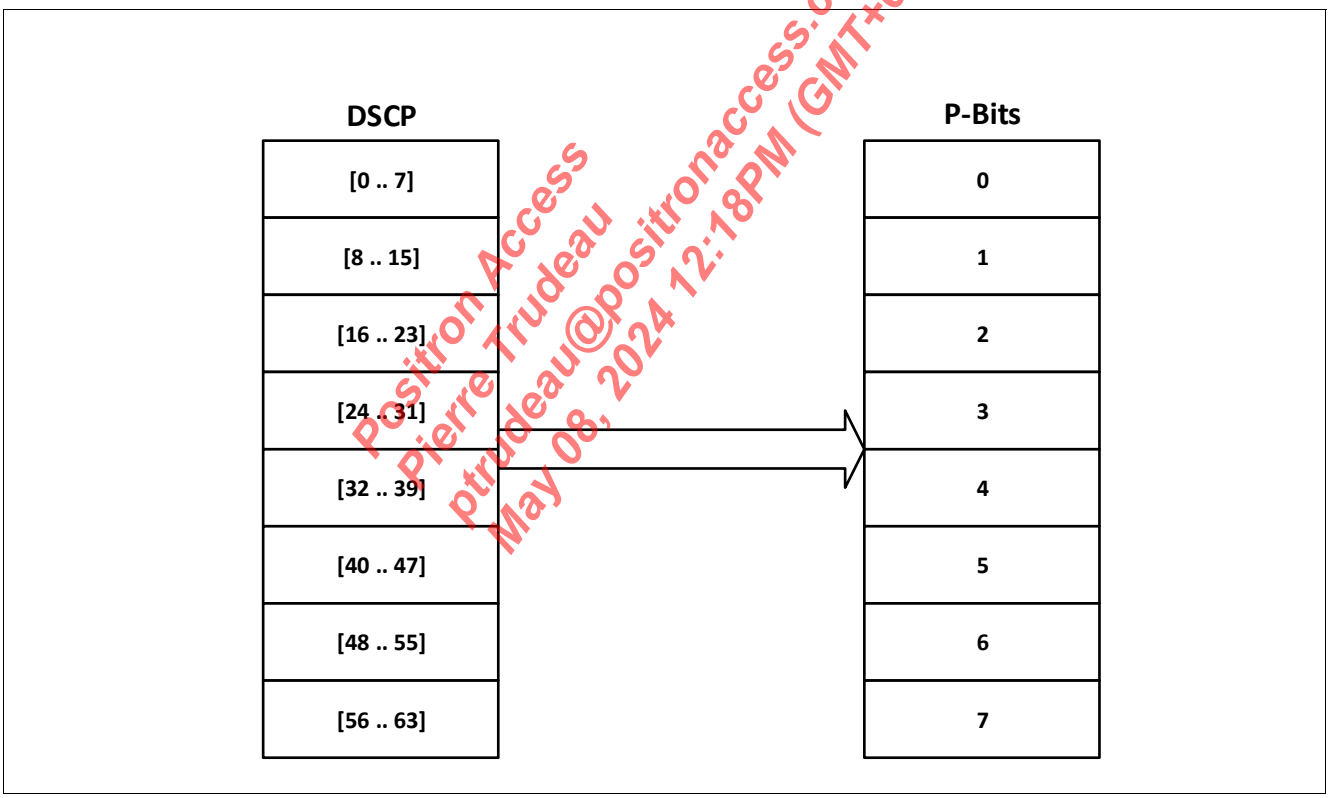


Figure 97 Extended VLAN DSCP to P-bit Mapping

The configuration of rules containing the P-bit DSCP derivation is achieved by using several rules. The maximum number is limited by the DSCP field size (64 different values). For **Figure 97**, these eight rules are used:

1. `tc filter add dev $eth ingress pref 1 protocol ip flower ip_tos 0xe0/0xe0 action vlan push id 333 priority 7 protocol 802.1q`
2. `tc filter add dev $eth ingress pref 2 protocol ip flower ip_tos 0xc0/0xc0 action vlan push id 333 priority 6 protocol 802.1q`
3. `tc filter add dev $eth ingress pref 3 protocol ip flower ip_tos 0xa0/0xa0 action vlan push id 333 priority 5 protocol 802.1q`

4. `tc filter add dev $eth ingress pref 4 protocol ip flower ip_tos 0x80/0x80 action vlan push id 333 priority 4 protocol 802.1q`
5. `tc filter add dev $eth ingress pref 5 protocol ip flower ip_tos 0x60/0x60 action vlan push id 333 priority 3 protocol 802.1q`
6. `tc filter add dev $eth ingress pref 6 protocol ip flower ip_tos 0x40/0x40 action vlan push id 333 priority 2 protocol 0x8100`
7. `tc filter add dev $eth ingress pref 6 protocol ip flower ip_tos 0x20/0x20 action vlan push id 333 priority 2 protocol 802.1q`
8. `tc filter add dev $eth ingress pref 7 protocol ip flower ip_tos 0x0/0x0 action vlan push id 333 priority 1 protocol 0x8100`

Special Cases Required for OMCI

The current Linux `tc flower` API must be extended or a new classifier for the hardware offloading must be introduced. The main effort is to add support for multiple VLANs (vlan depth). Another new feature is the copying of the priority bits from IEEE 802.1p-bits and DSCP headers. The kernel API must also be modified to allow usage of deprecated TPIDs, such as 0x9100 or 0x9200.

All necessary extensions for OMCI extended handling in Linux:

- Support for new/old TPIDs, such as 0x9100 or 0x9200. Currently Linux and `iproute2` only support IEEE 802.1q and IEEE 802.1ad.
- Support to the utilities and classifiers for multiple VLAN tags. OMCI requires up to 4 tag matching rules.
- Support for copying priority from inner/outer VLAN tags to new VLAN tags.
- Support for copying priority from DSCP to new VLAN tags.
- Support for TPID modifications different from IEEE 802.1q and IEEE 802.1ad (0x9100, 0x9200).

4.20.3.15 Dot1X Port Extension Package

ME class: 290

An instance of this managed entity is created by the OLT and represents a set of attributes that control a port's IEEE 802.1X operation. It is created and deleted autonomously by the ONU upon creation or deletion of a **PPTP Ethernet UNI**.

The implementation of this managed entity implies that the IEEE 802.1X "authenticator" is located within the OLT or upstream of the OLT, communicating with an "authentication server" such as a RADIUS server. The PPTP Ethernet UNI is the "controlled port" to be authenticated, while the "peer" is a station that is connected to this physical UNI port of the ONU. The system overview is illustrated in **Figure 98**.

Given this, the OLT is in charge of performing the authentication and to inform the ONU about the result. The ONU in turn allows or restricts data traffic across the related UNI port. If a UNI is set to the unauthenticated state, no packets other than EAPOL packets are exchanged across this interface.

This *includes* the access to an optional software server for telnet/SSH or a web server located within the ONU.

During OMCI initialization, authentication is disabled until the OLT intentionally activates it. To avoid network access before the authentication process has been successfully performed, the OLT must ensure that no T-CONT allocations or GEM/XGEM ports are configured for user traffic before the authentication is activated through the **Dot1X Enable** attribute.

Only a part of the attributes that are defined for this managed entity is used here. Other possible implementations, which include the authenticator entity within the ONU, will make use of the other attributes but are not covered by this implementation.

The implementation is for reference and must be enhanced by the system vendor.

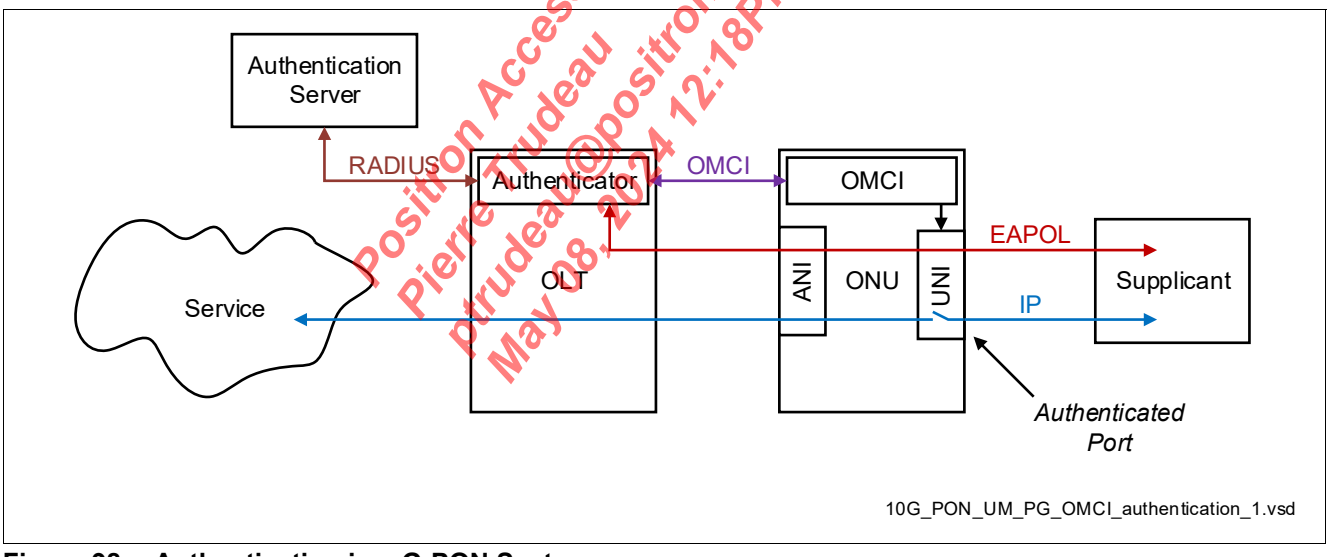


Figure 98 Authentication in a G-PON System

Managed Entity ID

2 bytes, read-only

This is a unique number for each instance of this managed entity. It is identical to the ID of the related **PPTP Ethernet UNI** managed entity.

Dot1X Enable

1 byte, read, write

This is a boolean read/write attribute that enables this managed entity as a whole. By default it is “false” and authentication is disabled. All traffic is allowed across the related **PPTP Ethernet UNI**. In this state, all other attributes are ignored.

Action Register

1 byte, write-only

This write-only attribute allows the OLT to control the authentication status of the Ethernet UNI.

Only two commands are used by the implementation, setting of other values is ignored (does not change the port state):

- 2: Force unauthenticated
- 3: Force authenticated

The authentication state is controlled by a setting of the LAN port. The parameter controls the status of the port:

- 0: Unauthenticated, all traffic is blocked in upstream and downstream, except for EAPOL packets.
- 1: Authenticated, all upstream and downstream traffic can pass.

Authenticator PAE State

1 byte, read-only

This attribute is not used by the implementation.

Backend Authentication State

1 byte, read-only

This attribute is not used by the implementation.

Admin Controlled Directions

1 byte, read, write

This attribute is not used by the implementation.

Operational Controlled Directions

1 byte, read-only

This attribute is not used by the implementation. It reports a value of 0 to indicate that authentication is applied to both data directions (upstream and downstream).

Authenticator Controlled Port Status

1 byte, read-only

This read-only attribute reports the current authentication state that has been set through the **Action Register** attribute.

- 1: Authenticated (authorized)
- 2: Not authenticated (unauthorized)

In case the authentication is disabled (**Dot1X Enable** is set to “false”), a status of “2” is reported.

Quiet Period

2 bytes, read, write

This attribute is not used by the implementation.

Server Timeout Period

2 bytes, read, write

This attribute is not used by the implementation.

Re-authentication Period

2 bytes, read-only

This attribute is not used by the implementation.

Re-authentication Enabled

1 byte, read-only

This attribute is not used by the implementation.

Key Transmission Enabled

1 byte, read, write

This attribute is not used by the implementation.

Actions

Get, set

Alarm Notifications

The following Alarms are defined for this managed entity:

#0: Local authentication allowed

#1: Local authentication denied

Alarms are not generated by the reference implementation.

4.20.3.16 Dot1 Rate Limiter

ME class: 298

This managed entity is created by the OLT and assigns rate limiters to different classes of upstream traffic (unicast, broadcast and multicast).

Figure 99/**Figure 100** show the managed entity relationship.

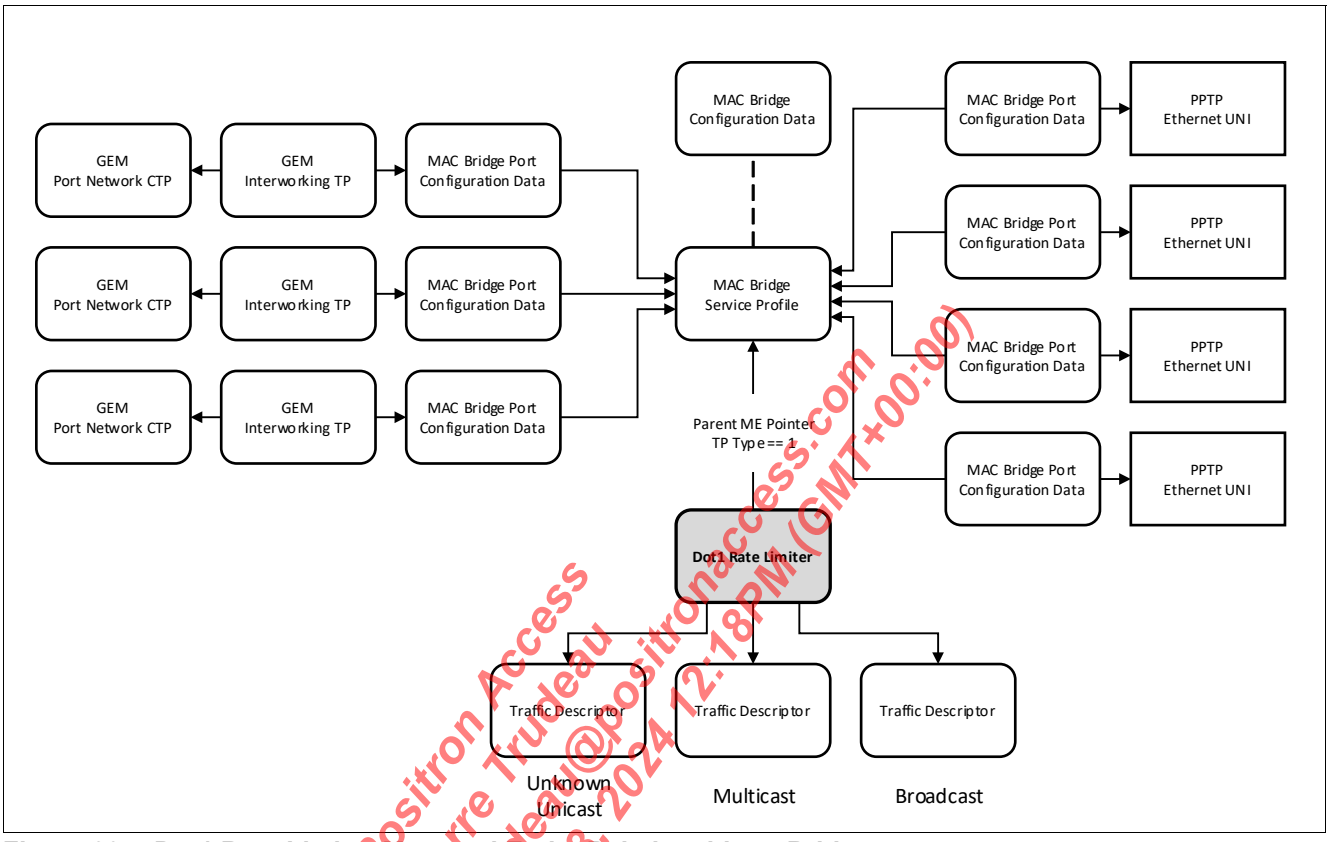


Figure 99 Dot1 Rate Limiter Managed Entity Relationship to Bridge

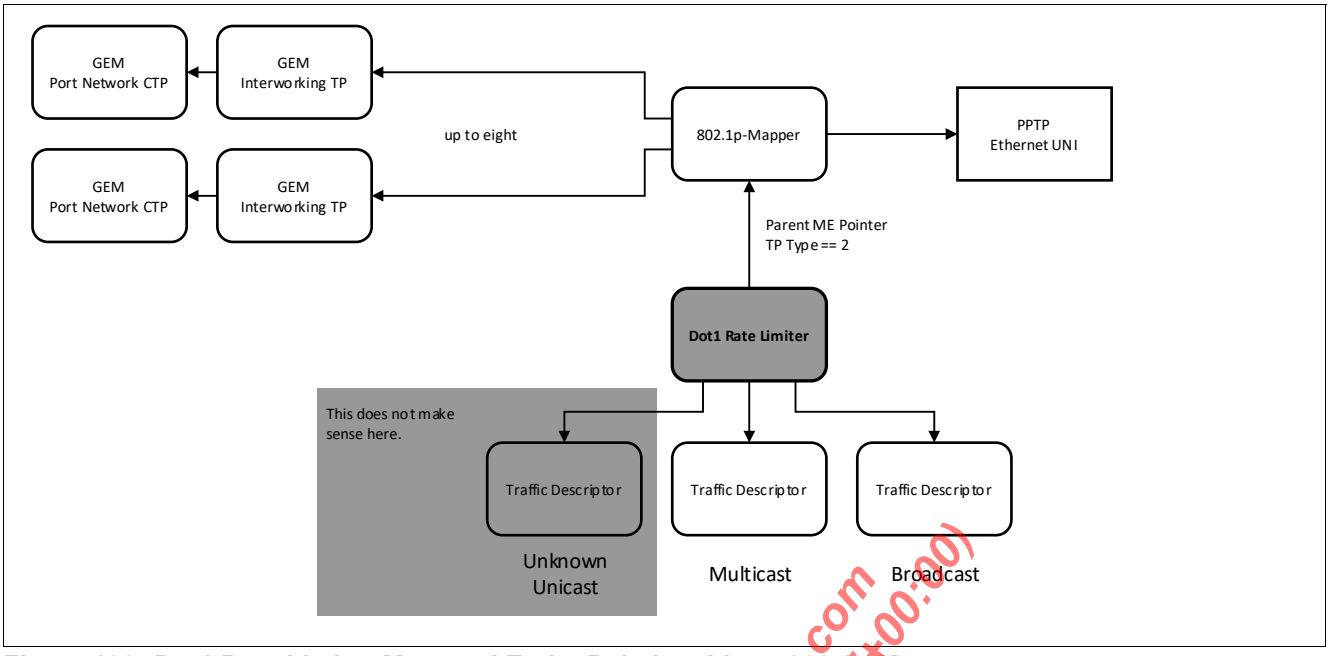


Figure 100 Dot1 Rate Limiter Managed Entity Relationship to 802.1p Mapper

Managed Entity ID

2 bytes, read-only, set-by-create

This is a unique number for each instance of this managed entity.

TP Type

2 bytes, read, write, set-by-create

This attribute defines the type of parent managed entity which is either a **MAC Bridge Service Profile** or a **IEEE 802.1p Mapper Service Profile**. This information is used to decide the implementation mode.

Parent ME Pointer

1 byte, read, write, set-by-create

This is the pointer to the parent managed entity.

Upstream Unicast Flood Rate Pointer

2 bytes, read, write, set-by-create

This attribute points to a **Traffic Descriptor** managed entity to select a policer that limits the upstream unicast traffic of flooded packets, due to unknown MAC destination address.

Follow the pointer to the implementation of the **Traffic Descriptor** and evaluate the meter index used for this specific managed entity.

Upstream Broadcast Rate Pointer

2 bytes, read, write, set-by-create

This attribute points to a **Traffic Descriptor** managed entity to select a policer that limits the upstream unicast traffic of broadcast packets, directed to the broadcast MAC destination address.

Follow the pointer to the implementation of the **Traffic Descriptor** and evaluate the meter index used for this specific managed entity.

Upstream Multicast Payload Rate Pointer

2 bytes, read, write, set-by-create

This attribute points to a **Traffic Descriptor** managed entity to select a policer that limits the upstream multicast traffic.

Follow the pointer to the implementation of the **Traffic Descriptor** and evaluate the meter index used for this specific managed entity.

Actions

Create, delete, get, set

Notifications

This managed entity does not generate notifications.

4.20.3.17 IEEE 802.1ag Maintenance

This chapter describes the managed entities provided to support IEEE 802.1ag Ethernet OAM CFM handling. Within OMCI, the term “dot1ag” is used to identify the related instances.

The listed managed entities are supported as “dummy implementation”, which means that attribute data is exchanged between ONU and OLT and locally stored. The related Ethernet OAM functions, however, are subject to a vendor-specific implementation.

These managed entities are provided as templates:

- **Dot1ag Maintenance Domain**
- **Dot1ag Maintenance Association**
- **Dot1ag Default MD Level**
- **Dot1ag MEP**
- **Dot1ag MEP Status**
- **Dot1ag MEP CCM Database**
- **Dot1ag CFM Stack**
- **Dot1ag Chassis-Management Info**

The relationship between the Dot1ag managed entities and the associated bridge or IEEE 802.1p mapper is shown in **Figure 101** and **Figure 102**, respectively.

Note: The software implementation does not cover the p-mapper-only configuration.

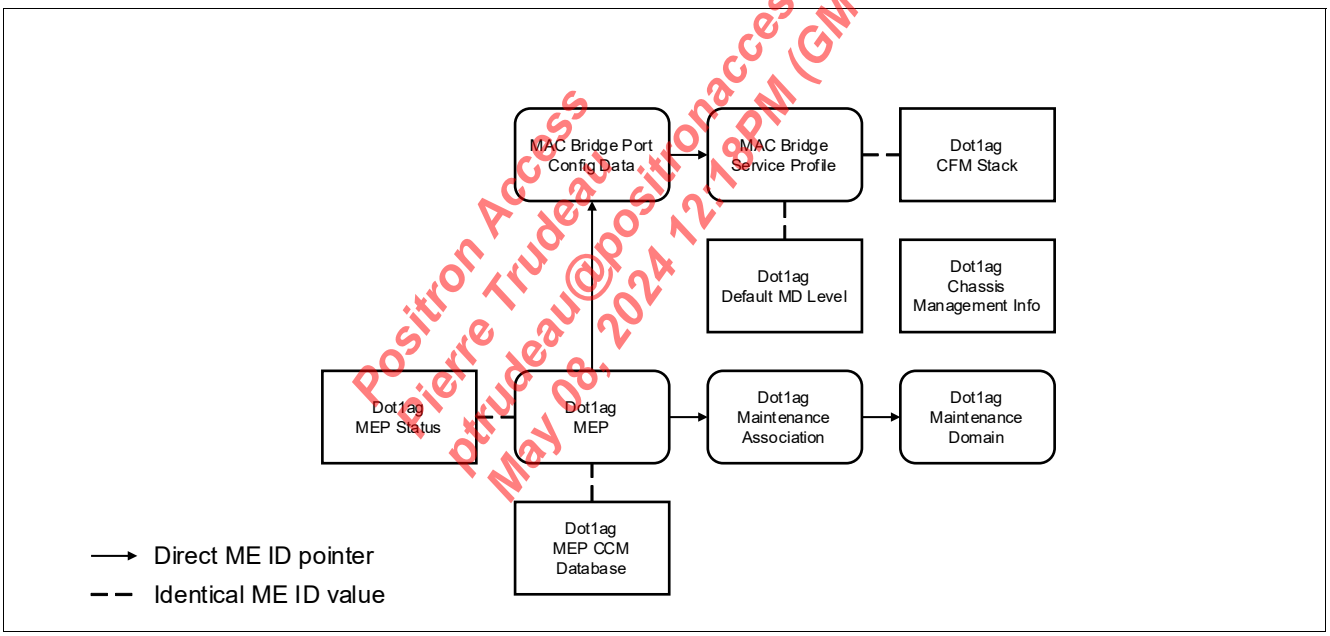


Figure 101 Dot1ag Relationship to a MAC Bridge

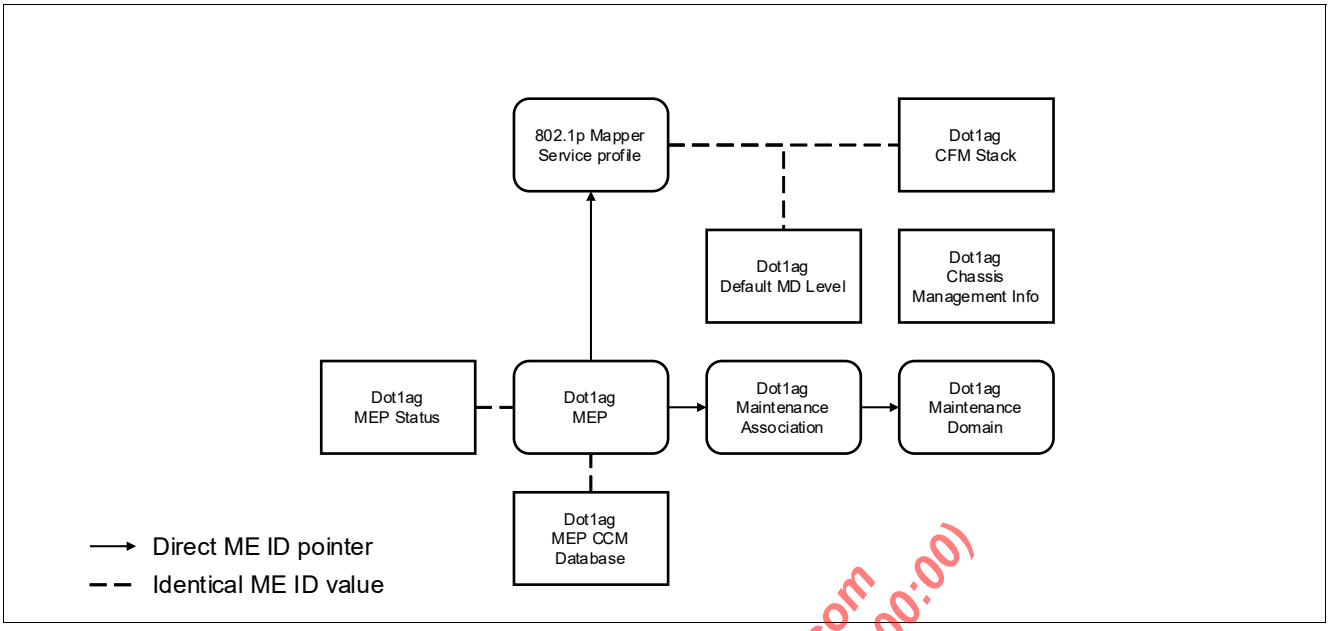


Figure 102 Dot1ag Relationship to an IEEE 802.1p Mapper

4.20.3.17.1 Dot1ag Maintenance Domain

ME class: 299

This managed entity is created and deleted by the OLT and provides configuration data for a Management Domain (MD). It is associated with a bridge or 802.1p mapper through indirect linkage.

Actions performed on this managed entity are create, delete, get, and set. There are no notifications.

Note: This managed entity is provided as a reference only, the functional implementation must be added by the system vendor.

Managed Entity ID

2 bytes, read-only, set-by-create

This is a unique number for each instance of this managed entity. The values 0 and 0xFFFF are reserved.

MD Level

1 byte, read, write, set-by-create.

Valid entries are 0 to 7.

MD Name Format

1 byte, read, write, set-by-create.

Valid entries are 1, 2, 3, 4, and 32.

MD Name 1

25 bytes, read, write.

Upon creation the value is set to all-zero by the ONU. A valid string must be set by the OLT after creation.

MD Name 2

25 bytes, read, write.

Upon creation the value is set to all-zero by the ONU. A valid string must be set by the OLT after creation.

MHF Creation

1 byte, read, write, set-by-create.

Valid entries are 1, 2, and 3.

Sender ID Permission

1 byte, read, write, set-by-create.

Valid entries are 1, 2, 3, and 4.

Actions

Create, delete, get, set

Notifications

This managed entity does not generate notifications.

4.20.3.17.2 Dot1ag Maintenance Association

ME class: 300

This managed entity is created and deleted by the OLT and models the service which is defined on a bridge port.

Actions performed on this managed entity are create, delete, get, and set. There are no notifications.

Note: This managed entity is provided as a reference only, the functional implementation must be added by the system vendor.

Managed Entity ID

2 bytes, read-only, set-by-create

This is a unique number for each instance of this managed entity. The values 0 and 0xFFFF are reserved.

MD Pointer

2 bytes, read, write, set-by-create.

Short MA Name Format

1 byte, read, write, set-by-create.

Valid entries are 1, 2, 3, and 4

Short MA Name 1

25 bytes, read, write.

Upon creation the value is set to all-zero by the ONU. A valid string must be set by the OLT after creation.

Short MA Name 2

25 bytes, read, write.

Upon creation the value is set to all-zero by the ONU. A valid string must be set by the OLT after creation.

CCM Interval

1 byte, read, write, set-by-create.

Associated VLANs

24 bytes, read, write.

This attribute provides an array of 12 entries, each of them 2 byte wide.

MHF Creation

1 byte, read, write, set-by-create.

Sender ID Permission

1 byte, read, write, set-by-create.

Valid entries are 1, 2, 3, and 4.

Actions

Create, delete, get, set

Notifications

This managed entity does not generate notifications.

4.20.3.17.3 Dot1ag Default MD Level

ME class: 301

This managed entity is created by the ONU automatically upon creation of a MAC bridge or of a stand-alone p-mapper which is not connected to a MAC bridge. The linkage is done through an identical managed entity ID value.

Actions performed on this managed entity are get, get next, and set. There are no notifications.

Note: This managed entity is provided as a reference only, the functional implementation must be added by the system vendor.

Managed Entity ID

2 bytes, read-only, set-by-create

Through an identical ID, this managed entity is implicitly linked to an instance of the **MAC Bridge Service Profile** managed entity or an IEEE **IEEE 802.1p Mapper Service Profile** managed entity. It is expected that an ONU will implement CFM on bridges or on IEEE 802.1p mappers, but not both, depending on its provisioning model. For precision, the reference is disambiguated by the value of the **Layer-2 Type** pointer attribute.

Layer-2 Type

1 byte, read-only

The value to be reported reflects the OMCI model which is used:

- 0: This managed entity is connected to a **MAC Bridge Port Configuration Data**.
- 1: This managed entity is connected to an **IEEE 802.1p Mapper Service Profile**.
  This option is not supported by the software.

Catchall Level

1 byte, read, write.

Valid entries are from 0 to 7.

The default level upon creation is defined by a `#define` value in the source code and can be adapted by the ONU vendor.

Catchall MHF Creation

1 byte, read, write.

Valid entries are 1, 2, and 3.

The default level upon creation is defined by a `#define` value in the source code and can be adapted by the ONU vendor.

Catchall Sender ID Permission

1 byte, read, write.

Valid entries are 1, 2, 3, and 4.

The default level upon creation is defined by a `#define` value in the source code and can be adapted by the ONU vendor.

Default MD Level Table

29 byte per table entry, read, write

The table entry fields are:

- Primary VLAN ID (2 byte)
- Table Control (1 byte)
- Status (1 byte)
- Level (1 byte)
- MHF Creation (1 byte)
- Sender ID Permission (1 byte)
- Associated VLANs List (22 byte)

Upon automatic creation, the attributes are empty (0). The attribute values must be defined by the external Ethernet OAM software.

Actions

Create, delete, get, set, set table

Notifications

This managed entity does not generate notifications.

4.20.3.17.4 Dot1ag MEP

ME class: 302

This managed entity is created and deleted by the OLT and models a Management Endpoint (MEP).

Actions performed on this managed entity are create, delete, get, set, and test.

Note: This managed entity is provided as a reference only, the functional implementation must be added by the system vendor.

Note: The test function is subject to future implementation

These alarms are defined, but not implemented in the software:

- 0: RDI CCM
- 1: MAC status
- 2: Remote CCM
- 3: Error CCM
- 4: Xcon CCM
- 5: Unexpected period
- 6: AIS

Managed Entity ID

2 bytes, read-only, set-by-create

This is a unique number for each instance of this managed entity.

Layer-2 Entity Pointer

2 bytes, read, write, set-by-create.

This is a pointer to either a **MAC Bridge Port Configuration Data** or an **IEEE 802.1p Mapper Service Profile** managed entity.

Layer-2 Type

1 byte, read, write, set-by-create.

This attribute defines the type of the Layer-2 Entity Pointer.

- 0: **MAC Bridge Port Configuration Data**
- 1: **IEEE 802.1p Mapper Service Profile**
  This option is not supported by the software.

MA pointer

2 bytes, read, write, set-by-create.

This is a pointer to a **Dot1ag Maintenance Association** managed entity.

MEP ID

2 bytes, read, write, set-by-create.

Valid entries are from 0 to 8191.

MEP Control

1 byte, read, write, set-by-create.

Primary VLAN

2 bytes, read, write, set-by-create.

Administrative State

1 byte, read, write.

Upon creation the state is set to “locked” (1). The OLT must unlock the managed entity function once the configuration is complete.

CCM and LTM Priority

1 byte, read, write, set-by-create.

Egress Identifier

8 bytes, read, write, set-by-create.

Peer MEP IDs

24 bytes, read, write.

Upon creation this attribute is set to all-zero.

ETH AIS Control

1 byte, read, write, set-by-create.

Fault Alarm Threshold

1 byte, read, write, set-by-create.

Alarm Declaration Soak Time

2 bytes, read, write.

Upon creation the value is set to the recommended default of 250 (2.5 s).

This value can be modified by changing a `#define` in the source code.

Alarm Clear Soak Time

2 bytes, read, write.

Upon creation the value is set to the recommended default of 1000 (10 s).

This value can be modified by changing a `#define` in the source code.

Actions

Create, delete, get, set, test

Alarm Notifications

The following alarms are defined for this managed entity:

#0: RDI CCM

#1: MAC status

#2: Remote CCM

#3: Error CCM

#4: Xcon CCM

#5: Unexpected period

#6: AIS

Alarms are not generated by the reference implementation.

4.20.3.17.5 Dot1ag MEP Status

ME class: 303

This managed entity provides read-only information related to a **Dot1ag MEP**. It is created or deleted autonomously by the ONU each time a **Dot1ag MEP** is created or deleted by the OLT.

Actions performed on this managed entity are get and get next. There are no notifications.

The values to be reported are defined by the external, vendor-specific user-space application which implements the MEP function. This application is *not* part of the SDK.

Note: This managed entity is provided as a reference only, the functional implementation must be added by the system vendor.

By default all values are read as 0.

Managed Entity ID

2 bytes, read-only, set-by-create

This is a unique number for each instance of this managed entity. Through an identical ID, this managed entity is implicitly linked to an instance of the **Dot1ag MEP** managed entity.

MEP MAC Address

6 bytes, read-only.

Fault Notification Generator State

1 byte, read-only.

Highest Priority Defect Observed

1 byte, read-only.

Current Defects

1 byte, read-only.

Last Received Errored CCM Table

N bytes (N not exceeding 128), read-only.

Last Received xcon CCM Table

N bytes (N not exceeding 128), read-only.

Out-of-Sequence CCMs Count

4 bytes, read-only.

CCM Transmitted Count

4 bytes, read-only

Unexpected LTR Count

4 bytes, read-only.

LBR Transmitted Count

4 bytes, read-only.

Next Loopback transaction Identifier

4 bytes, read-only.

Next Linktrace Transaction Identifier

4 bytes, read-only.

Actions

Get, get next

Notifications

This managed entity does not generate notifications.

4.20.3.17.6 Dot1ag MEP CCM Database

ME class: 304

This managed entity provides read-only information related to a **Dot1ag MEP**. It is created or deleted autonomously by the ONU each time a **Dot1ag MEP** is created or deleted by the OLT.

Actions performed on this managed entity are get and get next. There are no notifications.

The values to be reported are defined by the external, vendor-specific user-space application which implements the MEP function. This application is *not* part of the SDK.

Note: This managed entity is provided as a reference only, the functional implementation must be added by the system vendor.

By default all values are read as 0.

Managed Entity ID

2 bytes, read-only, set-by-create

This is a unique number for each instance of this managed entity. Through an identical ID, this managed entity is implicitly linked to an instance of the **Dot1ag MEP** managed entity.

Database Tables

There are 12 individual database tables provided to support up to 12 MEP instances:

- RMEP 1 Database Table
- …
- RMEP 12 Database Table

Each of these attributes can vary in length and provides the data fields:

- RMEP identifier (2 byte)
- RMEP state (1 byte)
- Failed-ok time (4 byte)
- MAC address (6 byte)
- RDI (1 byte)
- Port status (1 byte)
- Interface status (1 byte)
- Sender ID TLV (N byte)

Upon automatic creation, the tables are empty. The table contents must be defined by the external Ethernet OAM software.

Actions

Get, get next

Notifications

This managed entity does not generate notifications.

4.20.3.17.7 Dot1ag CFM Stack

ME class: 305

This managed entity is created by the ONU automatically upon creation of a MAC bridge or of a stand-alone p-mapper which is not connected to a MAC bridge. The linkage is done through an identical managed entity ID value.

Actions performed on this managed entity are get and get next. There is an AVC notification upon changes in the config error list table, this is for later implementation.

The values to be reported are defined by the external, vendor-specific user-space application which implements the MEP function. This application is *not* part of the SDK.

Note: This managed entity is provided as a reference only, the functional implementation must be added by the system vendor.

By default all values are read as 0.

Managed Entity ID

2 bytes, read-only, set-by-create

Through an identical ID, this managed entity is implicitly linked to an instance of the **MAC Bridge Service Profile** managed entity or an IEEE **IEEE 802.1p Mapper Service Profile** managed entity. It is expected that an ONU will implement CFM on bridges or on IEEE 802.1p mappers, but not both, depending on its provisioning model. For precision, the reference is disambiguated by the value of the **Layer-2 Type** pointer attribute.

Layer-2 Type

1 byte, read-only

The value to be reported reflects the OMCI model which is used:

- 0: This managed entity is connected to a **MAC Bridge Port Configuration Data**.
- 1: This managed entity is connected to an **IEEE 802.1p Mapper Service Profile**. *This option is not supported by the software.*

MP Status Table

N x 18 bytes, read-only.

These fields are provided per table entry:

- Port ID (2 byte)
- Level (1 byte)
- Direction (1 byte)
- VLAN ID (2 byte)
- MD (2 byte)
- MA (2 byte)
- MEP ID (2 byte)
- MAC address (6 byte)

Configuration Error List Table

N x 6 bytes, read-only

These fields are provided per table entry:

- VLAN ID (2 byte)
- Port ID (2 byte)
- Detected configuration error (1 byte)

Upon automatic creation, the tables are empty. The table contents must be defined by the external Ethernet OAM software.

Actions

Get, get next

Notifications

This managed entity generates the following AVC notification:

AVC #3: Config error list table

4.20.3.17.8 Dot1ag Chassis-Management Info

ME class: 306

The ONU autonomously creates an instance of this managed entity. The definition is done in the MIB initialization file. There is only one instance and the managed entity ID is set to 0.

Actions performed on this managed entity are get and set. There are no notifications.

Note: This managed entity is provided as a reference only, the functional implementation must be added by the system vendor.

Managed Entity ID

2 bytes, read-only, set-by-create

There is only one instance of this managed entity, the ID value is 0.

Chassis ID Length

1 byte, read-only.

Chassis ID Subtype

1 byte, read, write.

Chassis ID Part 1

25 bytes, read, write.

Chassis ID Part 2

25 bytes, read, write.

Management Address Domain Length

1 byte, read, write.

Management Address Domain 1

25 bytes, read, write.

Management Address Domain 2

25 bytes, read, write.

Actions

Get, set

Notifications

This managed entity does not generate notifications.

4.20.3.18 Multicast Operation

These managed entities are provided to support the configuration of multicast packet handling:

- **Multicast GEM Interworking Termination Point**
- **Multicast Operations Profile**
- **Multicast Subscriber Config Info**
- **Multicast Subscriber Monitor**

These managed entities are created by the OLT and control the handling of multicast data flows in downstream direction. The information given here is tightly connected with the multicast control protocol in use, which is IGMP for IPv4 or MLD when IPv6 is used. **Figure 103** to **Figure 106** depict examples of the possible relationship between these managed entities.

Figure 103 is an example of basic implementation using a single bridge for unicast and multicast traffic as well as IEEE 802.1p mapper.

Figure 105 shows that for separation of user traffic, separate bridges are used per UNI port.

Figure 106 depicts the possibility to configure individual bridges for unicast and multicast traffic.

It is possible for the **Multicast Operations Profile** (MOP) to be shared by all user ports or implemented per user port individually. It is also possible for multiple MOP entities to be used on a single UNI (per VLAN), when the multicast service package table of the **Multicast Subscriber Config Info** managed entity is used.

Figure 107 shows it is not possible to configure a IEEE 802.1p mapper-only structure because the multicast forwarding table of a bridge is internally used to filter the multicast group addresses.

The implementation details of these managed entities depend on the IGMP stack used, due to the information exchange between the OMCI managed entity attributes and IGMP. The description refers to the implementation example provided with the software. For vendor-specific IGMP/MLD implementation, it is possible for these to be used as an example but must be adapted according to the vendor individual implementation details.

Figure 103 shows an example provided with the software delivery that only covers the simple scenario. The support of more complex OMCI configurations must be covered by vendor-specific enhancements.

The reference software supports IGMP/MLD in native format (IPoE) for IPv4 and IPv6. It does not cover IGMP/MLD messages and multicast streams encapsulated into PPPoE tunnels.

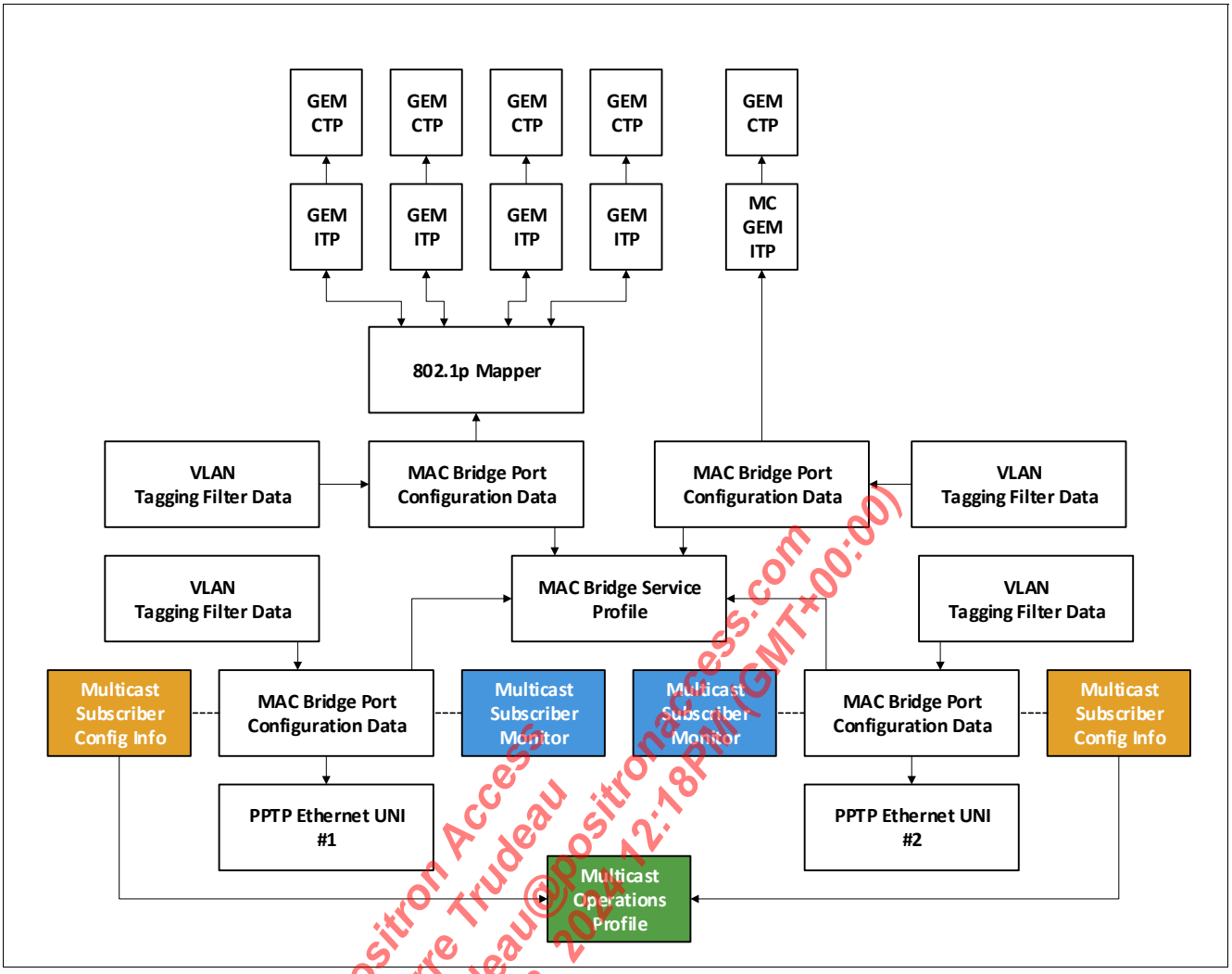


Figure 103 Multicast-related Managed Entities – Single Bridge Model – Single MOP

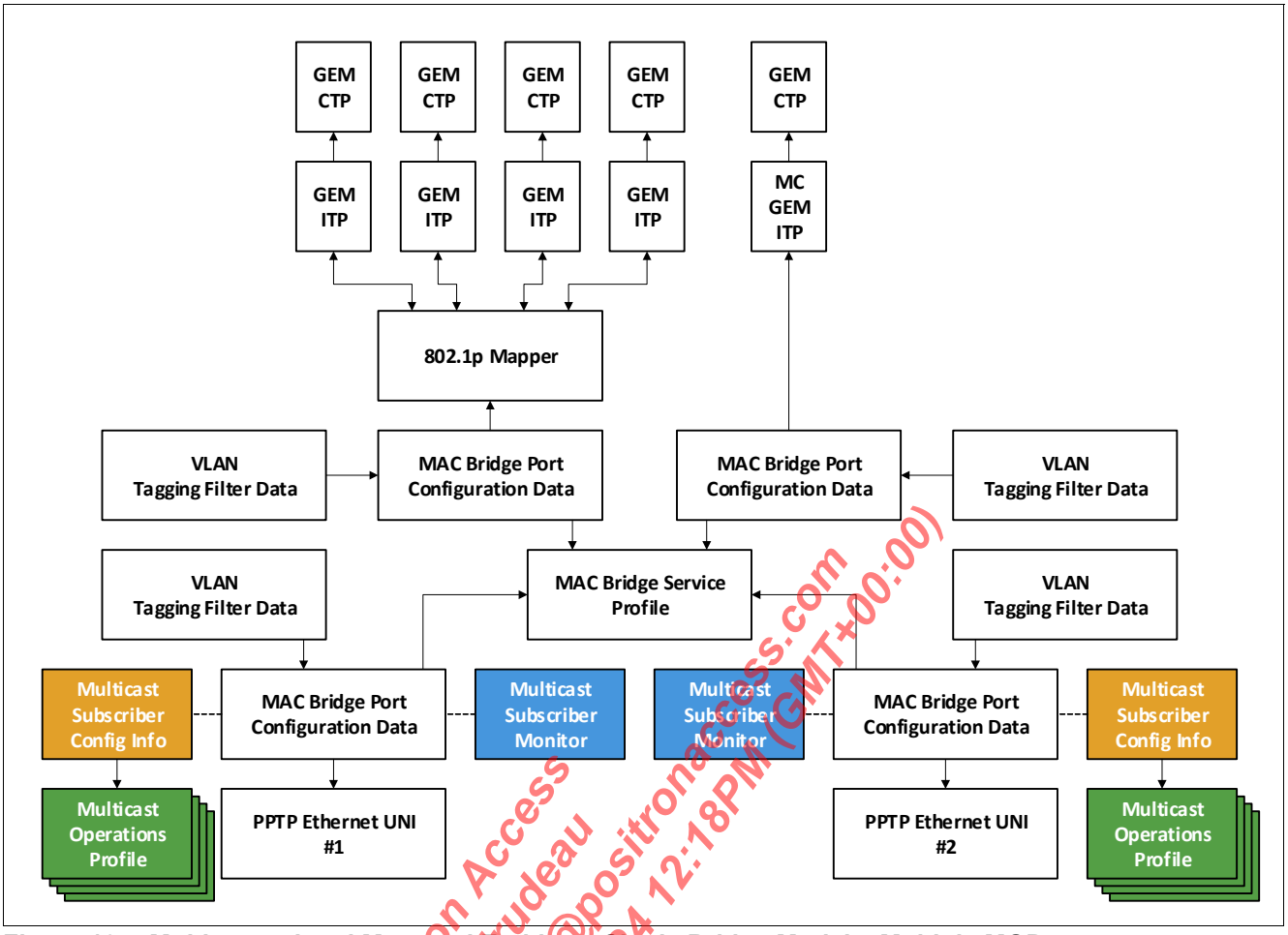


Figure 104 Multicast-related Managed Entities – Single Bridge Model – Multiple MOP

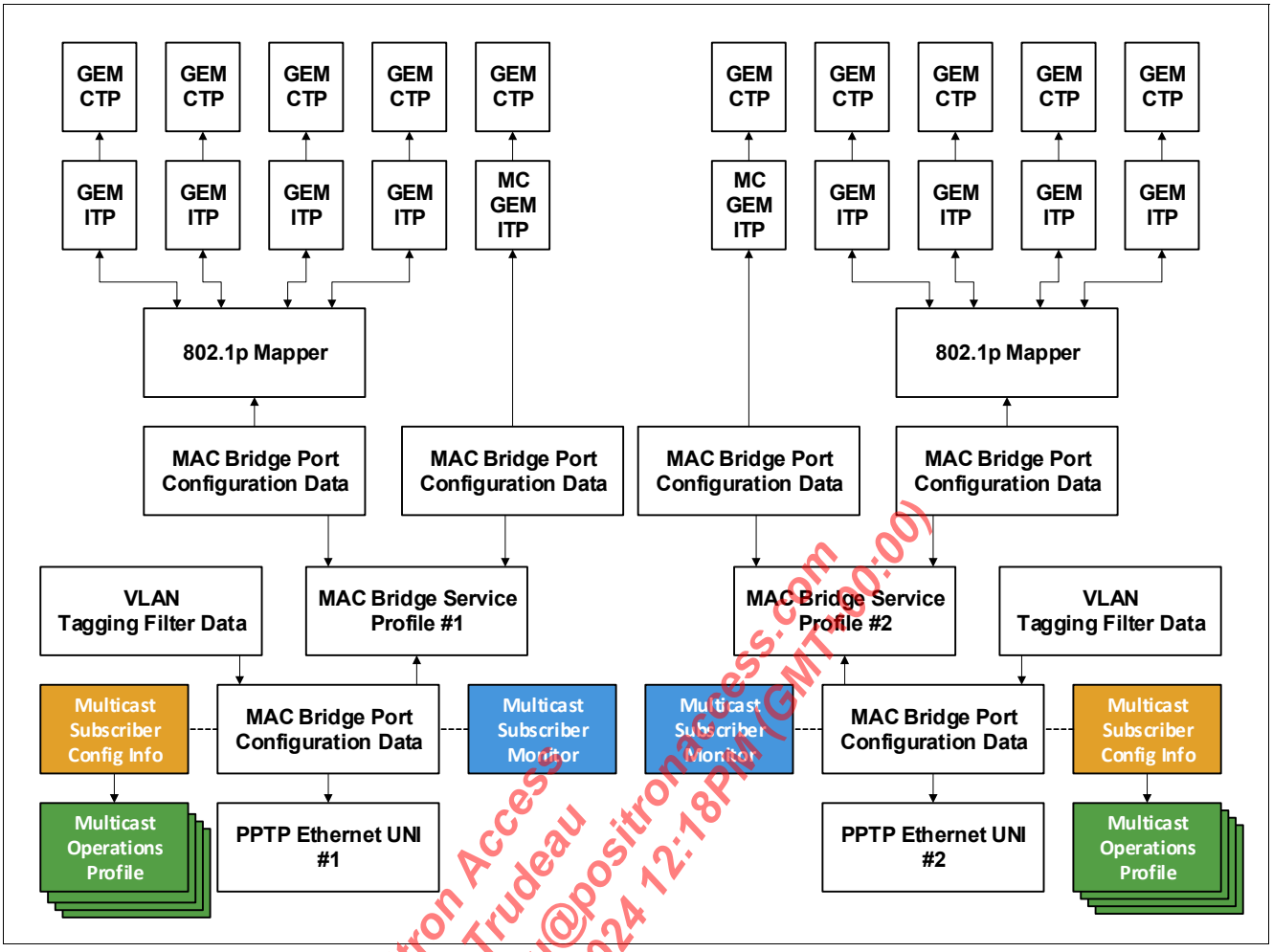


Figure 105 Multicast-related Managed Entities – Bridge and MOP per UNI Model

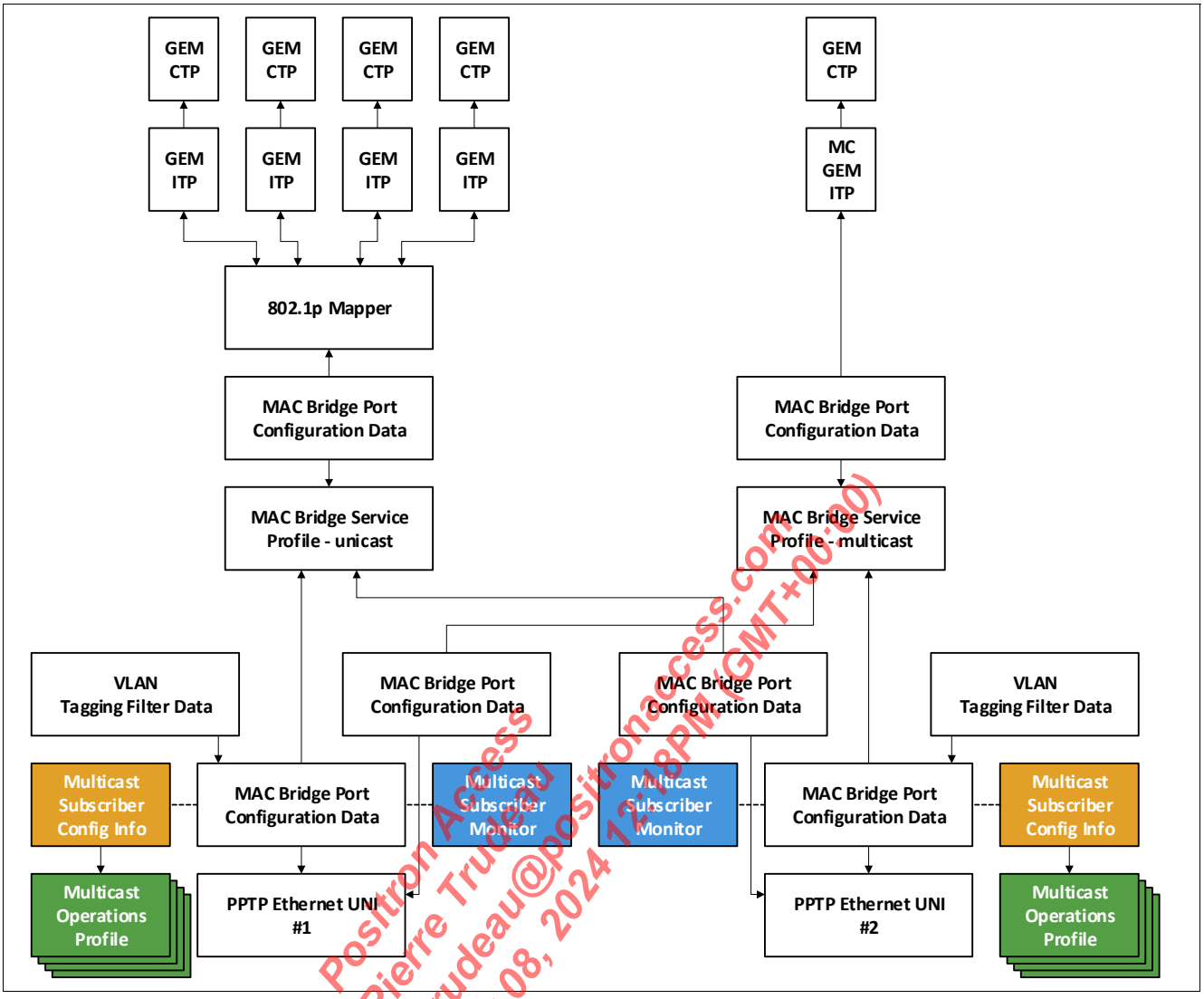


Figure 106 Multicast-related Managed Entities – Bridge per Traffic Model

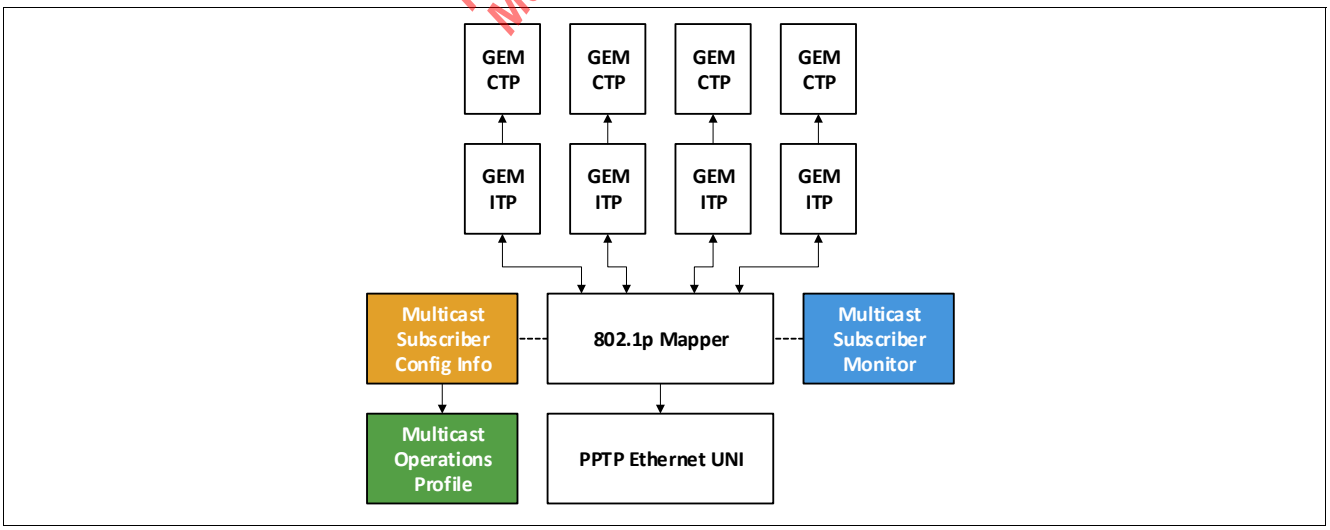


Figure 107 No Multicast without a Bridge

Note: When there is no bridge, packet multicasting is not supported.

IGMP and MLD Message Handling

To implement the multicast-related managed entities, the related protocol (IGMP or MLD) must be supported. **Figure 108** shows the IGMP or MLD control messages are received through a raw socket network interface. The hardware is configured to forward the IGMP/MLD control packets to the CPU by using a BPF packet filter in the kernel space. For details, check "*Creating a filter for IGMP or supported MLD types*" in `pon_net_mcc_core.c`.

The upstream IGMP/MLD packet insertion relies on the multicast VLAN tag to identify the upstream bridge port/GEM port which must be used.

Figure 109 shows the OMCI multicast daemon is able to handle the managed entities and IGMP/MLD control packets. The exact details of the handling of the three OMCI multicast managed entities **Multicast Operations Profile** (MOP), **Multicast Subscriber Config Info** (MSCI) and **Multicast Subscriber Monitor** (MSM) are described in the sections below. **Figure 109** also shows when the managed entity attributes must configure specific settings in the hardware, for example the VLAN tagging operation for multicast data traffic, updating the bridge MDB table, or multicast address filtering. The example stack supports the IGMP and MLD handling in snooping mode. Other modes, such as proxy mode, are up to the ONU vendor implementation.

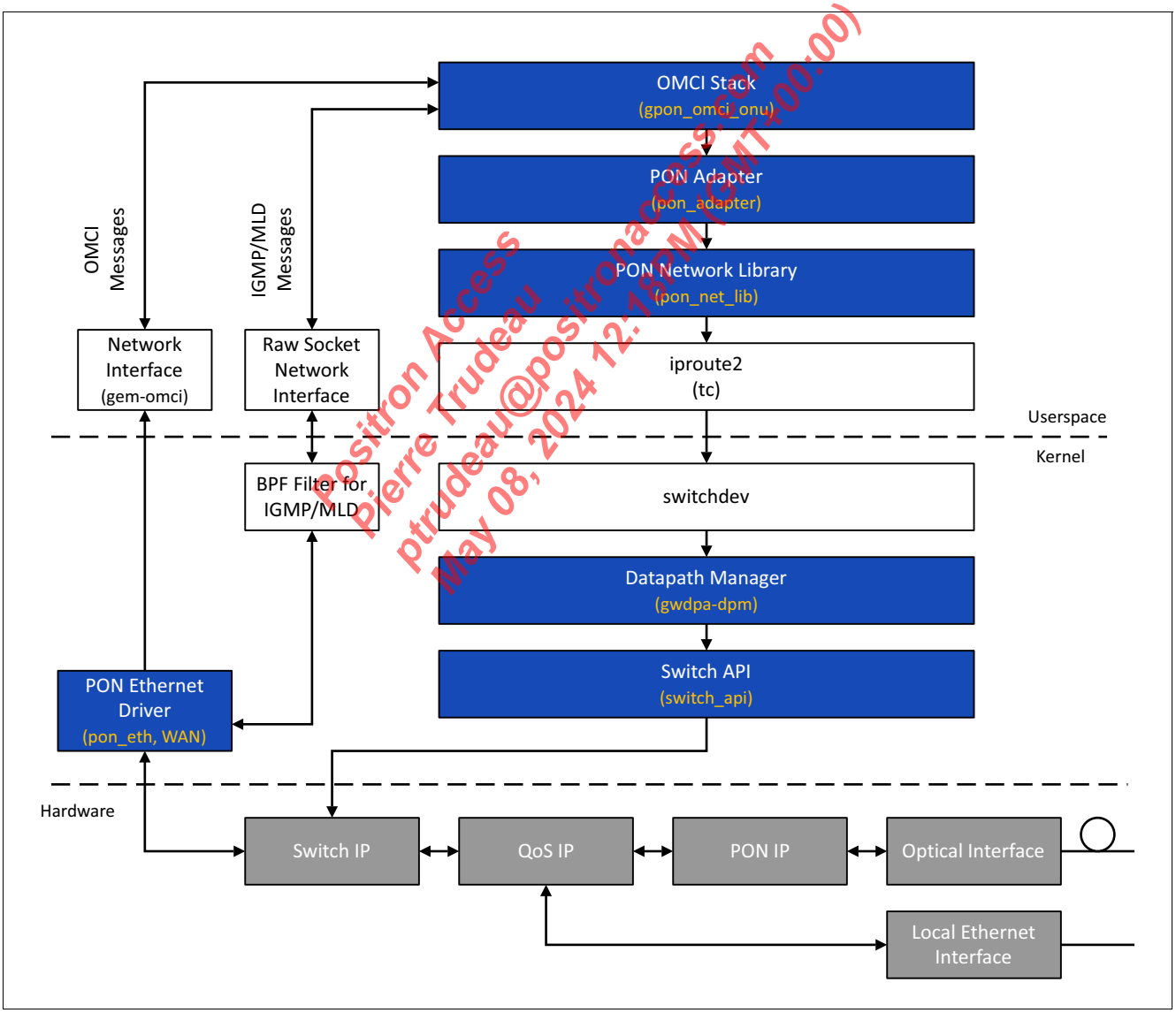


Figure 108 Multicast Software Layers

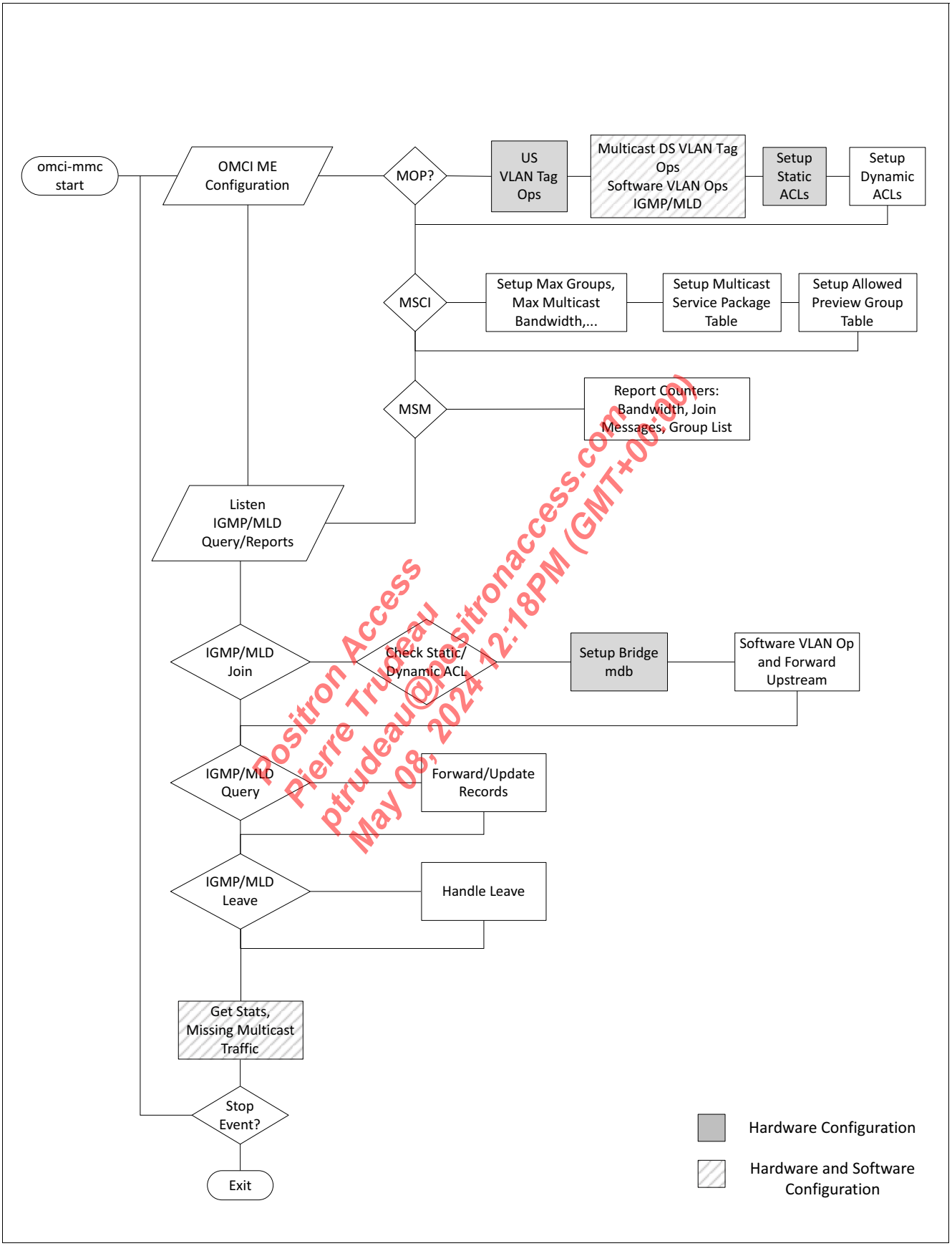


Figure 109 OMCI Multicast Handling Flowchart

4.20.3.18.1 Multicast Operations Profile

ME class: 309

This managed entity is created by the OLT and controls the multicast policy to be applied. The parameters defined by the OMCI through this managed entity are exchanged with the IGMP or MLD software layer to handle the multicast operation.

It is possible to apply the parameters defined through this managed entity to all multicast streams in common, individually per UNI port, or individually per VLAN and UNI port. This depends on the configuration of the **Multicast Subscriber Config Info** managed entity.

Managed Entity ID

2 bytes, read, write, set-by-create.

This attribute uniquely identifies this managed entity and is used by the **Multicast Subscriber Config Info** to link to it. The values are assigned by the OLT ranging from 0x001 to xFFFE.

IGMP Version

1 byte, read, write, set-by-create.

As the detailed functionality of this attribute is not specified by G.988, the implementation is done according to what is typically requested by the system vendors.

Pass the IGMP/MLD version to be used to the IGMP/MLD protocol handler, when it is one of the valid selections:

- 1: IGMPv1 – this is a deprecated code point, reject this setting.
  - When IGMP or MLD messages are received, these must be silently discarded.
- 2: IGMPv2
  - IGMPv2 messages are accepted and handled accordingly.
  - IGMPv3 messages are discarded and not handled.
  - MLDv1 messages are accepted and handled accordingly.
  - MLDv2 messages are accepted and handled accordingly.
- 3: IGMPv3
  - IGMPv2 messages are accepted and handled accordingly.
  - IGMPv3 messages are accepted and handled accordingly.
  - MLDv1 messages are accepted and handled accordingly.
  - MLDv2 messages are accepted and handled accordingly.
- 16: MLDv1
  - IGMPv2 messages are accepted and handled accordingly.
  - IGMPv3 messages are accepted and handled accordingly.
  - MLDv1 messages are accepted and handled accordingly.
  - MLDv2 messages are discarded and not handled.
- 17: MLDv2
  - IGMPv2 messages are accepted and handled accordingly.
  - IGMPv3 messages are accepted and handled accordingly.
  - MLDv1 messages are accepted and handled accordingly.
  - MLDv2 messages are accepted and handled accordingly.

Table 54 IGMP/MLD Message Handling

IGMP Message Type	Configured IGMP Version Attribute				
	IGMPv1[1)]	IGMPv2	IGMPv3	MLDv1	MLDv2
IGMPv1	Drop	Drop	Drop	Drop	Drop
IGMPv2	Drop	Handle	Handle	Handle	Handle

Table 54 IGMP/MLD Message Handling (cont'd)

IGMP Message Type	Configured IGMP Version Attribute				
	IGMPv1[1)]	**IGMPv2**	**IGMPv3**	**MLDv1**	**MLDv2**
IGMPv3	Drop	Drop	Handle	Handle	Handle
MLDv1	Drop	Handle	Handle	Handle	Handle
MLDv2	Drop	Handle	Handle	Drop	Handle

1) The selection is rejected.

IGMP Function

1 byte, read, write, set-by-create.

Pass the IGMP/MLD function to be used to the IGMP/MLD protocol handler, when it is one of the valid selections:

- 0: IGMP/MLD snooping[1)]
- 1: IGMP/MLD snooping with proxy reporting[2)]
- 2: IGMP/MLD proxy[2)]

Immediate Leave

1 byte, read, write, set-by-create.

Pass this control setting to the IGMP/MLD protocol handler:

- 0: Do not support immediate leave.
- 1: Support immediate leave.

Upstream IGMP TCI

2 bytes, read, write, set-by-create.

This defines the VLAN ID and p-bits to be added to the upstream IGMP/MLD messages, when requested by the **Upstream IGMP Tag Control** attribute.

Pass this setting to the IGMP/MLD protocol handler.

Upstream IGMP Tag Control

1 byte, read, write, set-by-create.

This attribute controls the upstream VLAN handling of IGMP/MLD messages. Pass this setting to the IGMP/MLD protocol handler:

- 0: Transparent forwarding.
  Used when there is a VLAN configuration defined by the **Extended VLAN Tagging Operation Configuration Data** managed entity. Otherwise, there is no adding, changing or removing of VLAN tag(s).
- 1: Add a VLAN tag as defined by the **Upstream IGMP TCI** attribute.
  The TPID to be used for an add operation is application dependent. Use one of the values dedicated for VLAN identification. See **Input TPID – Upstream** in **Section 4.20.3.14.2**.
- 2: Replace an existing (outer) VLAN tag completely (VID, DEI, and p-bit) by the value given through **Upstream IGMP TCI**. When none exists, add it as described under 1.
- 3: Replace an existing (outer) VLAN ID with the VID defined by the **Upstream IGMP TCI** attribute while maintaining the existing p-bits and DEI bit. When none exists, add one as defined by the **Upstream IGMP TCI** attribute, including the defined DEI and p-bits.

1) Covered by the example implementation.
2) Requires ONU vendor software enhancements.

To disable the extended VLAN operation on IGMP/MLD messages (and possible upstream multicast packets), the involved UNI ports must be configured accordingly, when this attribute is different from 0.

The handling of the VLAN operations for the IGMP/MLD control packets must be implemented in the software. The hardware must be configured not to interfere with the handling of these packets.

Depending on the selected attribute value, the software must implement these VLAN handling operations:

- 0: Do not modify the VLAN tag (when any).
- 1: Add an outer VLAN tag, with new VID, DEI and p-bits given by this attribute.
- 2: Replace the outer VLAN tag, with new VID, DEI, and p-bits given by this attribute.
- 3: Replace the outer VLAN tag, with new VID given by this attribute but keep DEI and p-bits.

Upstream IGMP Rate

4 bytes, read, write, set-by-create.

This attribute limits the IGMP/MLD control packet rate in the upstream direction. Pass this to the IGMP/MLD protocol handler. The value is given in packets/s. The control packets in excess of the given rate are dropped.

The rate limiter is implemented in the multicast control client (`mcc`) software.

To protect the software from an excessive IGMP/MLD flood, a hardware policer is in place to limit the total bit data rate of IGMP/MLD packets. The maximum allowed rate must be selected so that it is not below the value given by this attribute. The default hardware rate limit can be configured as described in **Chapter 3.15**, **Packet Rate Limitation to the CPU**.

Dynamic Access Control List Table

24 byte per row, read, write.

This table provides a list of multicast group address ranges. Each entry provides:

- Table Control
- GEM/XGEM port ID
- VLAN ID
- Source IP address (IPv4 or IPv6)
- Destination IP address, start of range
- Destination IP address, end of range
- Imputed group bandwidth
- Preview length
- Preview repeat time
- Preview repeat count
- Preview reset time

Pass this information to the IGMP/MLD protocol handler.

When checking for an entry or adding a new one, the PCP bits and DE bit of the outer VLAN (vlan_1) must be set to 0, the inner VLAN value (vlan_2) must always be set to 0.

Figure 111 shows the table format definition. This covers the IPv4/IGMP as well as IPv6/MLD implementations. The row part 0 is used by all the OMCI implementations while row parts 1 and 2 are only provided when the implementation follows the latest ITU-T G.988 versions.

Special handling is required for a VLAN ID value of 0 and those larger than 4095:

- VLAN ID = 0: This entry is related to an untagged multicast stream.
- VLAN ID = 4096: This value is reserved and must not be used.
- VLAN ID = 4097: This entry is related to a tagged multicast stream, but the VLAN ID value itself must be ignored. A join required on any VID for the related group (range) must be accepted.
- VLAN ID > 4097: These values are unspecified and must not be used.

Table Compatibility Handling

The table format has been changed in the course of OMCI evolution. While the newer table format provides three different rows, the initial table format only uses a single row. The row contents is the same for the first row. To allow the OLT to detect the table design in use, the first two bytes (table control) provide a Test bit. For the original type, the bit is set and read back as 1. For the updated version, a value of 0 is returned when the Test bit is read after a set operation.

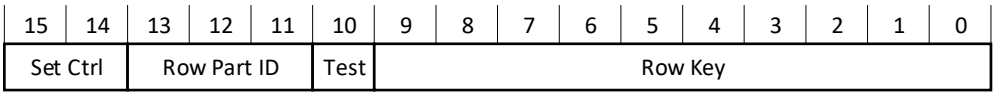


Figure 110 Table Control Word Structure

Set Control

These two bit control the meaning of a set operation. They are returned as 00_B upon a read access.

- 00_B: Reserved.
- 01_B: Write this entry into the table, overwrite an existing entry.
- 10_B: Delete this entry from the table (ignore the row part ID, Test bit, and row key value).
- 11_B: Clear all entries from the table (ignore the row part ID, Test bit, and row key value).

Row Part ID

The row part ID field distinguishes the row part associated with the current set or get operation. Row part 0 is backward compatible with earlier versions of this managed entity definition. Row parts 1-2 are optional on a row by row basis. They are set by using values 001_B to 010_B as the row part ID. Row parts 3 to 7 are reserved.

Byte	Row Part 0	Row Part 1	Row Part 2
1	Table Control	Table Control	Table Control
2			
3	GEM Port ID	IPv6 Source Address Leading Bytes	IPv6 Destination Address Leading Bytes
4			
5	VLAN ID (ANI)		
6			
7	Source IP Address		
8			
9			
10			
11	Destination IP Address, start of range		
12			
13			
14			
15	Destination IP Address, end of range	Preview Length	Reserved
16			
17		Preview Repeat Time	
18			
19	Imputed Group Bandwidth	Preview Repeat Count	
20			
21		Preview Reset Time	
22			
23	Reserved	Reserved	
24			

Figure 111 Dynamic Access Control List Table Format

Test

This bit allows the OLT to determine whether an ONU supports the extended format access control list. When the ONU does not support extended format, it is possible to set the test bit to 1 and read it back with a get and get next operation. When the ONU does support the extended format, this bit always returns the value 0 under a get next operation.

Row Key

The row key distinguishes rows in the table.

Static Access Control List Table

24 byte per row, read, write.

This attribute is a list that specifies one or more multicast group address ranges. The groups defined in this list are multicast on the associated UNI(s) unconditionally, without an IGMP join.

Pass this information to the IGMP/MLD protocol handler to configure the multicast streams beside the regular join/leave handling.

In addition to the addresses statically configured by the OLT, some well-known multicast addresses are always pre-configured in the multicast forwarding table to allow standard protocols to be handled in any case. See **Table 55**. Additions may be required to support other protocols by default without OLT configuration.

Table 55 Pre-configured Well-known IP Multicast Addresses

IP Protocol Version	IP Multicast Address	Direction	Usage
IPv4	224.0.0.1	Downstream	All hosts address to forward IGMP query messages in downstream
	224.0.0.2	Upstream	All routers address to forward IGMP reports (join/leave messages) in upstream
	224.0.0.22	Upstream Downstream	IGMPv3 group address to forward reports (join/leave messages) in upstream
IPv6	FF02::1	Downstream	All nodes address to forward MLD query messages in downstream
	FF02::2	Upstream	All routers address to forward MLD reports (join/leave messages) in upstream
	FF02::5	Upstream Downstream	OSPFv3
	FF02::6	Upstream Downstream	OSPFv3
	FF02::9	Upstream Downstream	RIP
	FF02::A	Upstream Downstream	EIGRP
	FF02::16	Upstream Downstream	All routers address to forward MLDv2 reports (join/leave messages) in upstream

The configuration of the static access control lists to the hardware must be done by using the Linux standard bridge APIs.

Lost Groups List Table

10 bytes per entry, read-only.

This attribute is a list of groups from the dynamic access control list table for which there is an active join, but no downstream flow is present. Each entry is a vector of these components:

- VLAN ID (0 when not used)
- Source IP address (0.0.0.0 when not used, least significant four bytes in case of IPv6)
- Multicast destination IP address (least significant four bytes in case of IPv6)

The OMCI multicast daemon must be able to monitor the activity for each multicast group.

The software must check all known multicast groups regularly to report those data streams, that did not arrive at the ONU.

In case a missing stream to a certain destination (group) is detected, the source IP address is reported as:

- 0.0.0.0 when no source address is available (not a source-specific multicast).
- The source address, when a single source address is related to the group address.
- A list of all source addresses, when multiple streams are booked from different source addresses to the same group address.

Because, in theory, there can be multiple streams sharing the same destination address but coming from different source addresses, the absence of a subset of streams cannot be detected.

While this table is not empty, the lost multicast group **Alarm Notifications** is activated.

Robustness

1 byte, read, write, set-by-create.

This attribute allows tuning for possible packet loss in the network. Pass this to the IGMP/MLD protocol handler.

Querier IP Address

4 bytes, read, write, set-by-create.

This attribute specifies the IP address to be used by a proxy querier. Pass this to the IGMP/MLD protocol handler.

Query Interval

4 bytes, read, write, set-by-create.

This attribute specifies the interval between general queries in seconds. Pass this to the IGMP/MLD protocol handler.

Query Maximum Response Time

4 bytes, read, write, set-by-create.

This attribute is the max response time added by the proxy into general query messages directed to UNIs. Pass this to the IGMP/MLD protocol handler.

Last Member Query Interval

4 bytes, read, write, set-by-create.

This attribute specifies the max response time inserted into group-specific queries sent to UNIs in response to group leave messages. It is also the repetition rate of [robustness] transmissions of the query. Pass this to the IGMP/MLD protocol handler.

Unauthorized Join Request Behavior

1 byte, read, write.

This boolean attribute specifies the reaction when an IGMP join request is received for a group not authorized in the dynamic address control list table, or an IGMPv3 membership report for groups, none of which are authorized in the dynamic ACL. The message is either forwarded or silently dropped, no join is entered into the forwarding table in any case. Pass this to the IGMP/MLD protocol handler.

Downstream IGMP and Multicast TCI

3 bytes, read, write, set-by-create.

This attribute is used to control the downstream VLAN handling of IGMP/MLD and multicast packets independent of the extended VLAN configuration. The attribute is composed of three bytes. The first byte defines the action to be taken, while the second and third byte contain VLAN information. The VLAN information is only required for some of the possible actions.

Depending on the selected attribute value, the corresponding VLAN treatment must be defined in the switch extended VLAN table by using the Linux `tc` APIs, as defined in **OMCI Extended VLAN Configuration in Linux**.

- 0: Transparent forwarding.
  Used when there is a VLAN configuration defined by the **Extended VLAN Tagging Operation Configuration Data** managed entity. Otherwise, there is no adding, changing or removing of VLAN tag(s).
- 1: Strip the outer VLAN tag.
- 2: Add an outer VLAN tag, with new VID, DEI and p-bits given by this attribute.
- 3: Replace the outer VLAN tag, with new VID, DEI, and p-bits given by this attribute.
- 4: Replace the outer VLAN tag, with new VID given by this attribute but keep DEI and p-bits.
- 5: Add an outer VLAN tag, with new VID, DEI and p-bits given by the **Multicast Subscriber Config Info** managed entity that is related.
- 6: Replace the outer VLAN tag, with new VID, DEI, and p-bits given by the **Multicast Subscriber Config Info** managed entity that is related.
- 7: Replace the outer VLAN tag, with new VID given by the related **Multicast Subscriber Config Info** managed entity, but keep the received DEI and p-bits.
- Other values: Return an error message, these are undefined code points.

The handling of the multicast traffic VLAN operation must be independent from the extended VLAN operations for the non-multicast traffic. The VLAN operations defined here must not interfere with the ones defined in **Extended VLAN Tagging Operation Configuration Data**. This is achieved by using different `tc` rules for the multicast packets filtering on the MAC address.

Actions

Create, delete, get, get next, set

Alarm Notifications

This alarm is defined for this managed entity:

#0: Lost multicast group

This alarm is activated as long as the number of entries in the **Lost Groups List Table** is not equal to zero.

4.20.3.18.2 Multicast Subscriber Config Info

ME class: 310

This managed entity is optionally created by the OLT.

The attributes provided by this managed entity must be forwarded to the IGMP/MLD protocol handler. When this managed entity is not present, all multicast join messages must be honored and no restriction is applied.

Managed Entity ID

2 bytes, read, write, set-by-create.

This attribute links this managed entity to a **MAC Bridge Port Configuration Data** and to a **Multicast Subscriber Monitor** managed entity by using an identical number.

Managed Entity Type

1 byte, read, write, set-by-create.

This attribute defines the relationship of the multicast subscriber configuration.

Only a **MAC Bridge Port Configuration Data** relationship is supported, because a bridge multicast table is required to hold the multicast information.

When the OLT tries to attach this managed entity to a **IEEE 802.1p Mapper Service Profile**, this request is rejected.

Multicast Operations Profile Pointer

2 bytes, read, write, set-by-create.

This is a pointer to the related **Multicast Operations Profile** managed entity. The use of this managed entity is an alternative configuration option for the use of the **Multicast Service Package Table** attribute provided within this managed entity.

*Note: This attribute is ignored, when the **Multicast Service Package Table** attribute is present (not empty).*

Maximum Simultaneous Groups

2 bytes, read, write, set-by-create.

This value limits the number of multicast groups allowed to be joined concurrently on the related user interface (bridge port). This value must be handed over to the IGMP/MLD software handling the join requests to limit the number of concurrent multicast groups accordingly. A value of 0 disables the limitation.

When the **Multicast Service Package Table** attribute is present, this value provides a global limit for the number on simultaneous multicast groups, in addition to the individual limits configured in the table. The global limit takes precedence when the sum of individual limits is greater.

Maximum Multicast Bandwidth

4 bytes, read, write, set-by-create.

This value limits the (imputed) bandwidth of all multicast groups allowed to be joined concurrently on the related user interface (bridge port). This value is given in units of byte/s and must be handed over to the IGMP/MLD software handling the join requests to limit the total bandwidth of the multicast groups accordingly. A value of 0 disables the limitation. The **Bandwidth Enforcement** attribute controls whether the join is done although the limit is already reached (or exceeded) or not.

A reference list of imputed bandwidth per multicast group is provided by the **Dynamic Access Control List Table** attribute of the **Multicast Operations Profile** managed entity.

When the **Multicast Service Package Table** attribute is present, this value provides a global limit for the total imputed bandwidth of simultaneous multicast groups, in addition to the individual limits configured in the table.

Bandwidth Enforcement

1 byte, read, write, set-by-create.

This attribute controls whether the attempts to exceed the given multicast group bandwidth limit must be accepted or denied. When false, the join is accepted but counted. When true, the join is rejected and counted.

The counter is provided by the **Bandwidth Exceeded Counter** attribute of the **Multicast Subscriber Monitor** managed entity.

Multicast Service Package Table

20 bytes per entry, read, write.

This attribute is a list specifying one or more multicast service packages. When present, the **Multicast Operations Profile Pointer** attribute is ignored.

When the ONU receives an IGMP/MLD join request, it searches the multicast service package table in row key order, matching the UNI-side VLAN ID (VID, as received on the UNI before any modification). Several rows may share the same VID. For each VID (UNI) match, the multicast operations profile pointer is used to access the managed entity containing the attributes associated with the service package. The search stops when all requested multicast groups are found and dealt with.

Each list entry is a vector of six components:

- Table control – 2 bytes
  The two MSBs (bit 15:14) of this field determine the meaning of a set operation.
  These bits are returned as 00_B during get next operations.
  – 00_B: Reserved.
  – 01_B: Writes this entry into the table. Overwrite any existing entry with the same row key.
  – 10_B: Deletes this entry from the table. The remaining fields are not meaningful.
  – 11_B: Clears all entries from the table. The remaining fields are not meaningful.
- VID (UNI) – 2 bytes
  The value in this field is compared with the VID of upstream IGMP/MLD messages, and is used to decide whether to honor a join request.
  – 0000_H to $0FFF_H$ (4095_D): Matches against the VID of the IGMP/MLD message. 0 indicates a priority-tagged message, whose P bits are ignored.
  – 1000_H (4096_D): Matches untagged IGMP/MLD messages only.
  – 1001_H (4097_D): Matches tagged messages only, but ignores the value of the VID.
  – 1002_H (4098_D) to $FFFF_H$: These values are unspecified and must never match.
- Maximum simultaneous groups – 2 bytes
  This field specifies the maximum number of dynamic multicast groups that may be replicated to the UNI-side bridge port at any one time, for the multicast service package associated with this row. The value 0 specifies that no administrative limit is to be imposed.
- Maximum multicast bandwidth – 4 bytes
  This field specifies the maximum imputed dynamic bandwidth, in bytes per second, that may be delivered to the UNI bridge port at any one time, for the multicast service package associated with this row. The value 0 specifies that no administrative limit is to be imposed.
  – The port is also constrained by the global **Maximum Simultaneous Groups** attribute and the **Maximum Multicast Bandwidth** attribute of this managed entity.
- Multicast operation profile pointer – 2 bytes
  This field contains the managed entity ID of the **Multicast Operations Profile** managed entity associated with this service package.
- Reserved – 8 byte
  Ignore on set operation, report 0 on get operation.

Allowed Preview Groups Table

22 bytes per entry, read, write.

This attribute is a list specifying the preview groups allowed for the UNI associated with this managed entity.

When an IGMP/MLD join request is received, the order of search precedence is as follows:

1. Multicast operations profile(s), fully authorized groups.
2. This attribute, the allowed preview groups table.
3. Multicast operations profile(s), preview-only groups.

When the first match is a group listed in this attribute, the ONU forwards the group to the UNI until the group is removed from this list, or until the subscriber leaves the group. The table structure is defined as:

- Table control – 2 bytes
  The two MSBs (bit 15:14) of this field determine the meaning of a set operation.
  These bits are returned as 00_B during get next operations.
  - 00_B: Reserved
  - 01_B: Write this entry into the table. Overwrite any existing entry with the same row key.
  - 10_B: Delete this entry from the table. The remaining fields are not meaningful.
  - 11_B: Clear all entries from the table. The remaining fields are not meaningful.
- The next lower three bits (bit 13:11) select one of the two row parts:
  - 000_B: The associated row operation is for row part 0.
  - 001_B: The associated row operation is for row part 1.
  - Others: Reserved.
- Bit 10 is reserved.
- The LSB part (bit 9:0) provides a row key, selecting the table row to be accessed.

Each row part provides a set of 22 bytes. The row parts are defined as:

- Row part 0, byte 0 and 1: Table control
- Row part 0, byte 2 to 17: Source IP address
  This field specifies the source IP address of the allowed preview group.
  Either an IPv4 address (first twelve bytes 0) or an IPv6 address.
- Row part 0, byte 18 and 19: ANI-side VLAN ID
  This field specifies the VLAN carrying the multicast group downstream.
  The VLAN ID resides in the 12 least significant bits; the remaining bits are set to 0 and not used.
  The value 0 designates an untagged downstream flow.
- Row part 0, byte 20 and 21: UNI-side VLAN ID
  This field specifies the VLAN carrying IGMP/MLD messages upstream across the UNI.
  The VLAN ID resides in the 12 least significant bits; the remaining bits are set to 0 and not used.
  The value 0 designates an untagged upstream flow.
- Row part 1, byte 0 and 1: Table control
- Row part 1, byte 2 to 17: Destination IP address
  This field specifies the destination IP address of the allowed preview group.
  Either an IPv4 address (first twelve bytes 0) or an IPv6 address.
- Row part 1, byte 18 and 19: Duration
  This field indicates the static length of time in minutes for which the group is authorized. The value 0 designates unlimited authorization.
- Row part 1, byte 20 and 21: Time left
  This field is controlled by the ONU (ignored during a set operation from OLT). It indicates how much time (measured in minutes) remains in the authorization. The ONU counts down; when this field reaches zero, the ONU deletes the entire entry from the table and stops replicating the group to the UNI. When the duration field specifies unlimited authorization, this field is ignored. The OLT may extend (or even truncate) the authorization by writing a new value into the duration field; the difference between new and old duration values is added to the time left field.

Actions

Create, delete, get, get next, set

Notifications

This managed entity does not generate notifications.

4.20.3.18.3 Multicast Subscriber Monitor

ME class: 311

This managed entity is created by the OLT and the attributes contained provide information that must be requested from the IGMP/MLD protocol handler.

Managed Entity ID

2 bytes, read, write, set-by-create.

This attribute links this managed entity to a **MAC Bridge Port Configuration Data** and to a **Multicast Subscriber Config Info** managed entity by using an identical number.

Managed Entity Type

1 byte, read, write, set-by-create.

This attribute defines the relationship of the multicast subscriber monitor.

Only a **MAC Bridge Port Configuration Data** relationship is supported, because a bridge multicast table is required to hold the multicast information.

When the OLT tries to attach this managed entity to a **IEEE 802.1p Mapper Service Profile**, this request is rejected.

Current Multicast Bandwidth

4 bytes, read-only.

This attribute provides the accumulation of the imputed multicast bandwidth of all multicast groups joined successfully.

A reference list of imputed bandwidth per multicast group is provided by the **Dynamic Access Control List Table** attribute of the **Multicast Operations Profile** managed entity.

The value is given in units of byte/s.

Join Messages Counter

4 bytes, read-only.

This 32-bit counter provides the number of accepted join events. When an IGMPv3 message attempts to join multiple channels, each channel is counted separately as if it had been joined through an individual message.

It rolls over to 0 when the maximum count is reached.

Bandwidth Exceeded Counter

4 bytes, read-only.

This 32-bit counter provides the number of attempts to exceed the granted maximum multicast bandwidth. The same handling applies for IGMPv3 as for the **Join Messages Counter**.

It rolls over to 0 when the maximum count is reached.

IPv4 Active Group List Table

24 bytes per entry, read-only.

This attribute lists the groups from the dynamic access control list table currently being actively forwarded, along with the actual bandwidth of each.

When a join is recognized from more than one IPv4 source address for a given group on this UNI, there is one table entry for each.

Table 56 lists the elements of a table entry.

Table 56 IPv4 Table Entry Format

Element	Size	Remarks
VLAN ID on UNI side	2 bytes	0 when not used
Source IP address	4 bytes	0.0.0.0 when not used
Multicast destination IP address	4 bytes	
Best efforts actual bandwidth estimate	4 bytes	Bytes per second
Client (set-top box) IP address	4 bytes	IP address of the device currently joined
Time since this client most recent join to the IP channel	4 bytes	In seconds
Reserved	2 bytes	

IPv6 Active Group List Table

58 bytes per entry, read-only

This attribute lists the groups from one of the related dynamic access control list tables or the allowed preview groups table currently being actively forwarded, along with the actual bandwidth of each.

When a join is recognized from more than one IPv6 source address for a given group on this UNI, there is one table entry for each.

In mixed IPv4-IPv6 scenarios, some fields may be IPv4, in which case their twelve most significant bytes of the given field are set to zero.

Table 57 lists the elements of a table entry.

Table 57 IPv6 Table Entry Format

Element	Size	Remarks
VLAN ID	2 bytes	0 when not used
Source IP address	16 bytes	0 when not used
Multicast destination IP address	16 bytes	
Best efforts actual bandwidth estimate	4 bytes	Bytes per second
Client (set-top box) IP address	4 bytes	IP address of the device currently joined
Time since this client most recent join to the IP channel	4 bytes	In seconds

Actions

Create, delete, get, get next, set

Notifications

This managed entity does not generate notifications.

4.20.4 Layer 3 Data Services

These managed entities are defined by ITU-T G.988 **[56]** in chapter 9.4.

4.20.4.1 IP Host Config Data

ME class: 134

This managed entity is created by the OLT and controls functions related to the termination of IP data or control traffic.

Multiple instances of this managed entity may be instantiated to support multiple software stacks, each using an individual IP address.

This type of managed entity is used to configure a software module identified by and accessed through an IPv4 layer 3 address.

For the lower layer network interface implementation, see **Section 3.16**.

Note: Only a single IP host network interface can be instantiated for 10G PON Chipset System Package 1.16 and below.

Managed Entity ID

2 bytes, read-only

This attribute uniquely identifies each instance of this managed entity. The ONU creates as many instances as there are independent IPv4 stacks on the ONU by definition in the MIB initialization file.

For easy discovery by the OLT, these managed entities should be numbered from 0 upwards. The ONU should create IP(v4) and IPv6 host config data managed entities with separate managed entity IDs, such that other managed entities can use a single TP type attribute to link with either.

IP Options

1 byte, read, write

This attribute is a bit map that enables or disables IP related options. The value 1 enables the option while 0 disables it. The default value of this attribute is 0.

- 01_H: Enable DHCP.
- 02_H: Respond to ping.
- 04_H: Respond to traceroute.
- 08_H: Enable the IP stack.
- 10_H: Reserved.
- 20_H: Reserved.
- 40_H: Reserved.
- 80_H: Reserved.

MAC Address

6 bytes, read-only

This attribute indicates the MAC address used by the IP node. The address is retrieved from the operating system network configuration of the host network device.

By default, the MAC address is automatically calculated from the U-Boot variable `ethaddr`. Under Linux, the MAC address is also reconfigured in `/etc/config/network`.

ONU Identifier

25 bytes, read, write

This is a unique ONU identifier string. When set to a non-null value, this string is used instead of the MAC address in retrieving DHCP parameters. When the string is shorter than 25 characters, it must be null terminated. Its default value is 25 null bytes.

IP Address

4 bytes, read, write

The address used for IP host services, this attribute default value is 0.

Mask

4 bytes, read, write

The subnet mask for IP host services, this attribute default value is 0.

Gateway

4 bytes, read, write

The default gateway address used for IP host services, this attribute default value is 0.

Primary DNS

4 bytes, read, write

The address of the primary DNS server, this attribute default value is 0.

Secondary DNS

4 bytes, read, write

The address of the secondary DNS server, this attribute default value is 0.

Current Address

4 bytes, read-only

Current address of the IP host service.

Current Mask

4 bytes, read-only

Current subnet mask for the IP host service.

Current Gateway

4 bytes, read-only

Current default gateway address for the IP host service.

Current Primary DNS

4 bytes, read-only

Current primary DNS server address.

Current Secondary DNS

4 bytes, read-only

Current secondary DNS server address.

Domain Name

25 bytes, read-only

When DHCP indicates a domain name, it is presented here. When no domain name is indicated, this attribute is set to a null string. When the string is shorter than 25 bytes, it must be null terminated. The default value is 25 null bytes.

Host Name

25 bytes, read-only

When DHCP indicates a host name, it is presented here. When no host name is indicated, this attribute is set to a null string. When the string is shorter than 25 bytes, it must be null terminated. The default value is 25 null bytes.

Relay Agent Options

2 bytes, read, write

This attribute is a pointer to a **Large String** managed entity whose content specifies one or more DHCP relay agent options.

Note: This attribute is not part of the reference implementation.

Actions

Get, set

Notifications

The following AVC notifications are defined for this managed entity:

AVC #9: Current IP address

AVC #10: Current network mask

AVC #11: Current gateway address

AVC #12: Current primary DNS server address

AVC #13: Current secondary DNS server address

AVC #14: Domain name

AVC #15: Host name

The reference implementation does not generate notifications for this managed entity.

4.20.4.2 IPv6 Host Config Data

ME class: 347

This managed entity is created by the OLT and controls functions that are related to the termination of IP data or control traffic.

Multiple instances of this managed entity can be instantiated to support multiple software stacks, each using an individual IP address.

This type of managed entity is used to configure a software module that is identified by and accessed through an IPv6 layer-3 address.

A template implementation is available for reference, the functional implementation is left open for a vendor-specific implementation.

Managed Entity ID

2 bytes, read-only

This attribute uniquely identifies each instance of this managed entity. The ONU creates as many instances as there are independent IPv4 stacks on the ONU by definition in the MIB initialization file.

For easy discovery by the OLT, these managed entities should be numbered from 0 upwards. The ONU should create IP(v4) and IPv6 host config data managed entities with separate managed entity IDs, such that other managed entities can use a single TP type attribute to link with either.

IP Options

1 byte, read, write

This attribute is a bit map that enables or disables IPv6 related options. The value 1 enables the option while 0 disables it. The default value of this attribute is 0.

- 01_H: Unlock the IPv6 administrative stack.
- 02_H: Enable Router Solicitation (RS)
- 04_H: Enable DHCPv6
- 08_H: Respond to ICMPv6 echo requests (IPv6 ping) with ICMPv6 echo replies.
- 10_H: Reserved
- 20_H: Reserved
- 40_H: Reserved
- 80_H: Reserved

MAC Address

6 bytes, read-only

This attribute indicates the MAC address used by the IPv6 node.

ONU Identifier

25 bytes, read, write

This is a unique ONU identifier string. If set to a non-null value, this string is used instead of the MAC address in retrieving DHCPv6 parameters. If the string is shorter than 25 characters, it must be null terminated. Its default value is 25 null bytes.

The remaining attributes are exchanged with the IPv6 stack implementation of the Linux operation system.

IPv6 Link Local Address

16 bytes, read-only

IPv6 Address

16 bytes, read, write

Default Router

16 bytes, read, write

Primary DNS

16 bytes, read, write

Secondary DNS

16 bytes, read, write

Current Address Table

N x 24 bytes, read-only

Current Default Router Table

N x 16 bytes, read-only

Current DNS Table

N x 16 bytes, read-only

DUID

25 bytes, read-only

On-link Prefix

16 bytes, read, write

Current On-link Prefix Table

N x 26 bytes, read-only

Relay Agent Options

2 bytes, read, write

Actions

Get, get next, set, test

Notifications

The following AVC notifications are defined for this managed entity:

AVC #9: Current IP address table

AVC #10: Current default router table

AVC #11: Current DNS table

AVC #14: Current on-link prefix table

The reference implementation does not generate notifications for this managed entity.

4.20.4.3 IP Host Performance Monitoring History Data

ME class: 135

This managed entity is created by the OLT and provides counters that are related to the termination of IP data or control traffic (IPv4 or IPv6). Each managed entity of this type is related to one **IP Host Config Data** managed entity.

There are no hardware counters involved, the counters must be implemented in software. Implementation details are specific to the IP stack that is used.

This managed entity is *not* part of the SDK delivery.

Managed Entity ID

2 bytes, res-only, set-by-create

This attribute uniquely identifies each instance of this managed entity. Through an identical ID, this managed entity is implicitly linked to an instance of the **IP Host Config Data** or **IPv6 Host Config Data** managed entity.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** managed entity that is pointed to by this value provides alarm thresholds which are used for the monitoring items. The **Threshold Data 2** managed entity is not used.

ICMP Errors

4 bytes, read-only

This attribute counts ICMP errors received.

DNS Errors

4 bytes, read-only

This attribute counts DNS errors received.

DHCP Time-outs

2 bytes, read-only

This attribute counts DHCP time-outs.

IP Address Conflict

2 bytes, read-only

This attribute is incremented whenever the ONU detects a conflicting IP address on the network. A conflicting IP address is one that has the same value as the one currently assigned to the ONU.

Out of Memory

2 bytes, read-only

This attribute is incremented whenever the ONU encounters an out of memory condition in the IP stack.

Internal Error

2 bytes, read-only

This attribute is incremented whenever the ONU encounters an internal error condition such as a driver interface failure in the IP stack.

Actions

Create, delete get, set, get current data

Notifications

The following threshold crossing alerts are defined for this managed entity:

TCA #1: IPNPM ICMP error

TCA #2: IPNPM DNS error

TCA #3: DHCP timeout

TCA #4: IP address conflict

TCA #5: Out of memory

TCA #6: Internal error

The reference implementation does not generate notifications.

4.20.4.4 TCP/UDP Config Data

ME class: 136

The TCP/UDP config data managed entity is created by the OLT and configures TCP- and UDP-based services that are offered from an IP host. No hardware programming is involved.

This managed entity is available as a reference implementation, to be enhanced by the system vendor.

Managed Entity ID

2 bytes, read-only, set-by-create

This attribute uniquely identifies each instance of this managed entity. It is recommended that the managed entity ID be the same as the TCP/UDP port number in the **Port ID** attribute.

Port ID

2 bytes, read, write, set-by-create

This attribute specifies the port number that offers the TCP/UDP service.

Protocol

1 byte, read, write, set-by-create

This attribute specifies the protocol type as defined by IANA.

TOS/DiffServ Field

1 byte, read, write, set-by-create

This attribute specifies the value of the TOS/diffserv field of the IPv4 header. The contents of this attribute may contain the type of service as defined by IETF RFC 2474 or a Differentiated Services Code Point (DSCP).

Valid values for DSCP are as defined by IANA.

IP Host Pointer

2 bytes, read, write, set-by-create

This attribute points to the **IP Host Config Data** managed entity associated with this TCP/UDP data.

Any number of ports and protocols may be associated with an IP host.

Actions

Create, delete, get, set

Notifications

This managed entity does not generate notifications.

4.20.4.5 TCP/UDP Performance Monitoring History Data

ME class: 342

This managed entity is created by the OLT and collects performance monitoring data related to a TCP or UDP port. There are no hardware counters involved, the counters must be implemented in software.

Note: This managed entity is not part of the reference implementation.

Managed Entity ID

2 bytes, read-only, set-by-create

This attribute uniquely identifies each instance of this managed entity. Through an identical ID, this managed entity is implicitly linked to an instance of the **TCP/UDP Config Data** managed entity.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** managed entity that is pointed to by this value provides three alarm thresholds which are used for the Lost Packets, Misinserted Packets, and Impaired Blocks. The **Threshold Data 2** managed entity is not used.

Socket Failed

2 bytes, read-only

This attribute is incremented when an attempt to create a socket associated with a port fails.

Listen Failed

2 bytes, read-only

This attribute is incremented when an attempt by a service to listen for a request on a port fails.

Bind Failed

2 bytes, read-only

This attribute is incremented when an attempt by a service to bind to a port fails.

Accept Failed

2 bytes, read-only

This attribute is incremented when an attempt to accept a connection on a port fails.

Select Failed

2 bytes, read-only

This attribute is incremented when an attempt to perform a select on a group of ports fails.

Actions

Create, delete get, set, get current data

Notifications

The following threshold crossing alerts are defined for this managed entity:

TCA #1: Socket failed

TCA #2: Listen failed

TCA #3: Bind failed

TCA #4: Accept failed

TCA #5: Select failed

The reference implementation does not generate notifications.

4.20.5 Ethernet Services

These managed entities are defined by ITU-T G.988 **[56]** in chapter 9.5.

- **PPTP Ethernet UNI**
- **Virtual Ethernet Interface Point**
- **Ethernet Performance Monitoring History Data**
- **Ethernet Performance Monitoring History Data 2**
- **Ethernet Performance Monitoring History Data 3**
- **Ethernet Frame Performance Monitoring History Data Upstream**
- **Ethernet Frame Performance Monitoring History Data Downstream**
- **Ethernet Frame Extended Performance Monitoring History Data**
- **Ethernet Frame Extended Performance Monitoring History Data 64-bit**

The performance counter managed entities are provided as defined by ITU, but not all are used at the same time by an OLT.

4.20.5.1 PPTP Ethernet UNI

ME class: 11

The Physical Path Termination Point (PPTP) Ethernet UNI managed entity controls functions related to a physical Ethernet LAN port (UNI).

This type of managed entity is created by the ONU itself, up to `ONU_GPE_LAN_MAX` instances are supported. The creation process sets up the internal structure and relationship between a UNI port, an (internal) egress port, a traffic scheduler and a set of up to eight **Priority Queue** instances.

*Note: Even when only a single priority queue is attached to a UNI port, a traffic scheduler must be set up in between due to the internal hardware architecture. This scheduler is **not** reflected in the OMCI structure.*

The basic configuration of a UNI port requires these steps:

1. Activate the related GPE/ICTRLL<n> hardware module (<n> is the UNI port index, ranging from 0 to `ONU_GPE_LAN_MAX - 1`).
2. Assign a logical egress port instance to the physical UNI port.
3. Assign a scheduler to the egress port.
4. Assign up to eight priority queues to the scheduler.

Managed Entity ID

2 bytes, read-only

This attribute uniquely identifies each instance of this managed entity and is defined in the MIB initialization file. Each instance is used to create a dedicated network interface instance (for example, eth0_0, eth0_1, etc.)

The first byte (MSB) is ignored, only the second byte (MSB) is used to distinguish the related network interfaces.

Expected Type

1 byte, read, write

This attribute reflects the type of the Ethernet interface expected when a pluggable circuit pack is used in a cardholder. In a fixed ONU application, this must be set to type 0 (autosense).

Sensed Type

1 byte, read-only

This attribute reflects the type of inserted Ethernet interface when a pluggable circuit pack is used in a cardholder. In a fixed ONU application, this must report either type 24 (10/100BASE-T) or type 47 (10/100/1000/…BASE-T), depending on the application UNI port configuration.

An internal connection (VEIP) must be reported as type 48, a debug-only interface (LCT) as type 39.

Auto Detection Configuration

1 byte, read, write

When the Ethernet interface uses an integrated Ethernet PHY, the default configuration must be set to 00_H (auto rate, any duplex mode). When the OLT changes the setting to a fixed rate (10, 100, or 1000 Mbit/s) in full duplex mode, this must be accepted and the PHY must be configured accordingly.

When an external Ethernet PHY is connected to this interface, the default setting and the allowed modifications depend on the external PHY capabilities. When the setting is changed, the PHY must be reconfigured through the external MDIO interface. To simplify the configuration, it is recommended to use MaxLinear GPHY devices which are identical to the integrated PHYs and handled by the same commands.

Ethernet Loopback Configuration

1 byte, read, write

This attribute controls the data loopback behind the Ethernet PHY of the UNI (see **Figure 112**). The internal PHYs can be directly controlled. The external PHYs are supported as well, when a MaxLinear PHY type is used.

The control of other externally connected Ethernet PHY devices may be implemented by the application software through the MDIO read/write functions.

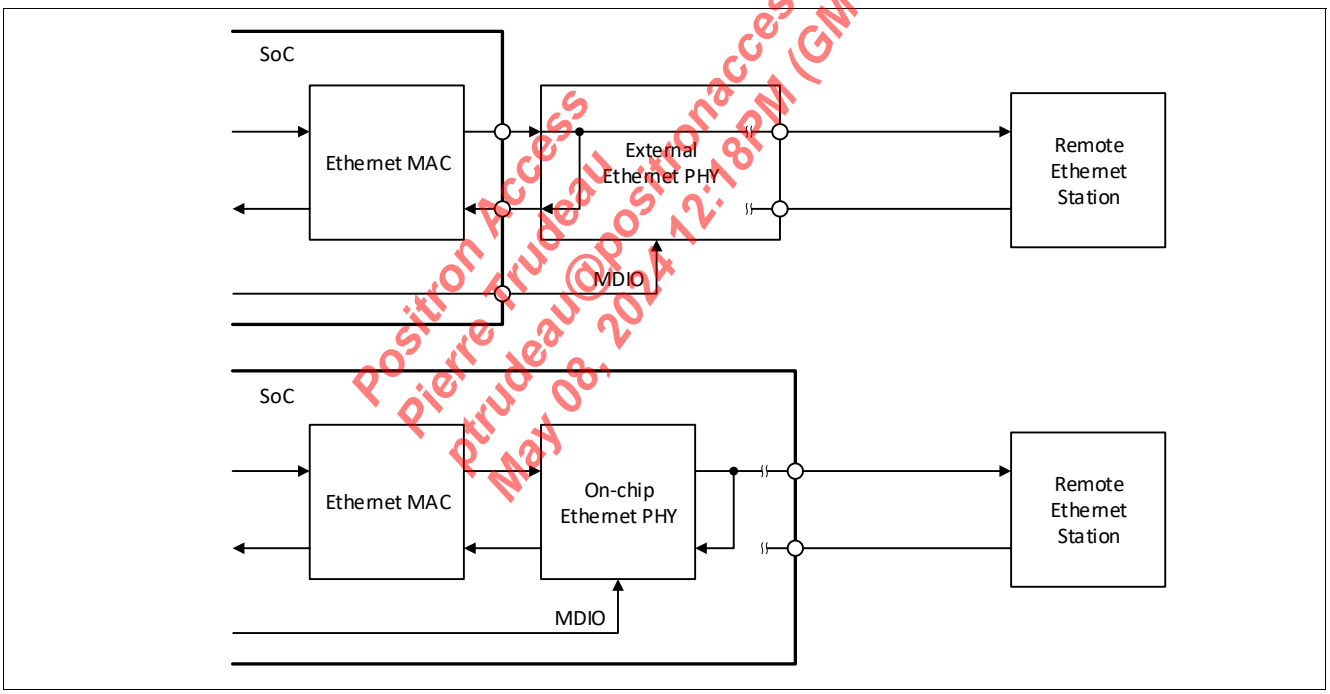


Figure 112 Ethernet Loopback

There is another option to enable a LAN side loopback within the Ethernet MAC layer and even swap the Ethernet MAC addresses (source versus destination). This is controlled by the ethtool function, using the option of private flags. The following flags are provided:

- `local_loopback_mac_enable`
- `local_loopback_mac_swap`

Command line example to read the status:

```
root@prx321-sfu-qspi-pon:/# ethtool --show-priv-flags eth0_0
Private flags for eth0_0:
bp_to_cpu_enable          : off
```

```
local_loopback_mac_enable : off
local_loopback_mac_swap   : off
qos_class_select_spcp_dscp: on
```

Administrative State

1 byte, read, write

This attribute locks (disables) or unlocks (enables) the function of the related interface.

- 0: The interface is unlocked and is able to transmit data
- 1: The interface is locked, no data transmission is possible.

The related **UNI-G** managed entity also provides an **Administrative State** attribute. An interface must be enabled only when both of them are set to unlocked.

Note: This rule may be violated by some OLTs, in this case, enable the interface based on the OLT requirements.

Special Handling on Single-Port ONU

When only a single Ethernet port is preset in the ONU and the LCT function is enabled to provide remote access through the UNI interface, the locking of the administrative state is handled differently.

When the LCT-enabled single port must be locked, the interface itself is not disabled but the internal connection between the UNI port and the related bridge port is cut. This ensures that no packets are sent to the upstream. The packets addressed to the LCT interface are handled as exceptions and forwarded as usual to the local network interface instance.

Under Linux, the change is made in `omcid.sh` by setting `lct = ""`.

Operational State

1 byte, read-only

This attribute indicates whether the managed entity is capable of performing its function. It is set to disabled (1) during device initialization. After the interface is initialized and able to connect to a remote station, the state is set to enabled (0). This state is maintained even when no physical connection is available and the link is down.

Configuration Indication

1 byte, read-only

This attribute reports the link configuration established by the PHY. The status is read from the Ethernet PHY through the MDIO interface. These PHY status bits must be checked:

- Register SR, bit LS: When the link is down, an unknown status is reported.
- Register MIISTAT, bit SPEED: Indicates the link speed (10, 100, or 1000 Mbit/s).
- Register MIISTAT, bit DPX: Indicates the duplex mode (only the full duplex mode is supported).

Maximum Frame Size

2 bytes, read, write

This attribute defines the maximum Ethernet frame size (also referred to as MTU) accepted upon reception and sent in transmit direction. When not defined for the given application, the default value of 1534 must be used.

The largest frame size to be handled is 9216 byte (jumbo frames).

Note: For the BBF TP-247 interoperability test case 6.6.1, this value is set to 4095 by the OLT emulator.

DTE or DCE Indication

1 byte, read, write

This is a fixed setting defined by the application hardware structure. It is relevant only for ports that do not include a PHY (use the default of 2 in this case, which means automatic detection). A parallel interface in PHY mode is

indicated as DCE (delivering the clock) while a parallel interface in MAC mode is indicated as DTE (receiving the clock).

Pause Time

2 bytes, read, write

The pause time used by the Ethernet MACs in case of in-band flow control is not configurable. The maximum value of $FFFF_H$ is always used and must be indicated when this attribute is read by the OLT.

Bridged or IP Indication

1 byte, read, write

This attribute always returns the code point 0 (bridged).

PPPoE Filter

1 byte, read, write

This attribute controls filtering of PPPoE packets on this Ethernet port.

The value 0 allows packets of all types. The value 1 discards everything but PPPoE packets.

The default value is 0.

Note: PPPoE filtering is not implemented.

Power Control

1 byte, read, write

The implementation of power over Ethernet is external to the CPE Software Suite device and beyond the scope of this software package. GPIO pins must be used to control this functionality in a vendor-specific way.

By default, this value is fixed to 0.

Disable Packet-based Time Synchronization

1 byte, read, write

This attribute controls the packet based time synchronization provided to an external equipment over the Ethernet (for ONUs capable of supporting packet-based time synchronization through the Precision Time Protocol managed entity and/or Synchronous Ethernet managed entity).

Note: This attribute is implemented. Time synchronization and Synchronous Ethernet are not under OMCI control.

Actions

Get, set

Attribute Value Change Notifications

An attribute type sense change is reported as an AVC event, sending the code point value of the detected type.

In the AVC implementation, this is different from the definition given in the **Configuration Indication**, which uses additional code points:

- 1: 10BASE-T
- 2: 100BASE-T
- 3: 1000BASE-T
- 4: 10GBASE-T

A change of the **Operational State** and an ARC timer expiration are also reported.

AVC #2: Sensed type

AVC #6: Operational state

AVC #12: ARC timer expiration

Alarm Notifications

This alarm is defined for this managed entity:

- LAN loss of signal
  - The implementation checks whether the LAN link is up.
    An event `ONU_EVENT_LAN<n>_LOS_ALARM_SET` is issued by the ONU driver. The alarm is cleared by the `ONU_EVENT_LAN<n>_LOS_ALARM_CLEAR` event.
    The value of <n> = 0, 1, 2, 3 identifies the affected LAN port.

4.20.5.2 Virtual Ethernet Interface Point

ME class: 329

This managed entity controls functions that are related to a Virtual Ethernet Interface Point (VEIP).

This type of managed entity is created by the ONU and is used to provide an interface to a different management domain. For example, gateway functions can be handled by the TR-069 (CWMP) software implemented on the ONU.

A dedicated egress queue, directed to the software, is used for the VEIP. The software creates a dedicated Ethernet interface for the VEIP. Only a single VEIP instance can be created.

Managed Entity ID

2 bytes, read, write, set-by-create.

This attribute uniquely identifies an instance of this managed entity type.

When used independently of a cardholder and circuit pack, the ONU should assign IDs in the sequence 1, 2, … When used in conjunction with a cardholder and circuit pack, this 2-byte number indicates the physical position of the VEIP. The first byte is the slot ID. The second byte is the port ID, with the range of 1 … 255.

The values 0 and 0xFFFF are reserved.

The value is assigned in the MIB initialization file and must be selected to match the OLT's expectation.

Administrative State

1 byte, read, write

This attribute locks (disables) or unlocks (enables) the function of the related virtual interface.

- 0: The interface is unlocked and can transmit data
- 1: The interface is locked, no data transmission is possible.

Operational State

1 byte, read-only

This attribute indicates whether or not the managed entity is capable of performing its function.

Valid values are:

- 0: Enabled
- 1: Disabled

Interdomain Name

25 bytes, read, write

This attribute is a character string that provides an optional way to identify the virtual Ethernet interface point to a non-OMCI management domain. The interface may also be identified by its managed entity ID, IANA assigned port, and possibly other ways. If the vendor offers no information in this attribute, it should be set to a sequence of 25 null bytes in the MIB initialization file.

TCP/UDP Pointer

2 bytes, read, write

This is a pointer to a TCP/UDP Config Data managed entity which provides for OMCI management of the non-OMCI management domain's IP connectivity. If no OMCI management of the non-OMCI domain's IP connectivity is required, this attribute may be omitted or set to its default, a null pointer.

IANA Assigned Port

2 bytes, read-only

This attribute contains the TCP or UDP port value as assigned by IANA for the management protocol associated with this virtual Ethernet interface.

This attribute is to be regarded as a hint, not as a requirement that management communications use this port; the actual port and protocol are specified in the associated TCP/UDP config data managed entity. If no port has been assigned, or if the management protocol is free to be chosen at run-time, this attribute should be set to 0xFFFF.

Actions

Get, set

Attribute Value Change Notifications

AVC #2: Operational state

Alarm Notifications

This alarm is defined for this managed entity:

- Connection function fail
  The implementation is application-specific.

4.20.5.3 Power over Ethernet Control

ME class: 349

This managed entity is implemented as a reference template only. The functional implementation must be added for applications that support PoE output ports.

This managed entity is automatically created through the MIB initialization if the ONU supports OMCI-controlled PoE functionality.

Managed Entity ID

2 bytes, read-only

The managed entity ID is the same as for the related PPTP Ethernet UNI.

PoE Capabilities

2 bytes, read-only

This is a 16-bit read-only value, only the LSB is functionally defined. Other bits are reserved and shall be set to 0.

- Bit 0 = 0: The PSE pin-out alternative is fixed.
- Bit 0 = 1: The PSE pin-out alternative is configurable by the OLT.

Power Pair Pin out Control

1 byte, read, write

This attribute is used to configure and report the pin-out configuration.

Operational State

1 byte, read

Valid values are enabled (0) and disabled (1).

Power Detection Status

1 byte, read

Power Classification Status

1 byte, read

Power Priority

1 byte, read, write

Invalid Signature Counter

2 bytes, read

Power Denied Counter

2 bytes, read

Overload Counter

2 bytes, read

Short Counter

2 bytes, read

MPS Absent Counter

2 bytes, read

PSE Class Control

1 byte, read, write

Current Power Consumption

4 bytes, read-only

Note: This attribute is not covered by the reference implementation.

Actions

Get, set

Attribute Value Change Notifications

AVC #3: Operational state

4.20.5.4 Ethernet Performance Monitoring History Data

ME class: 24

This managed entity is created by the OLT and provides Ethernet-related performance data. The associated interface is defined by the PPTP Ethernet UNI managed entity with an identical managed entity ID. The managed entity relationship is illustrated in **Figure 113**.

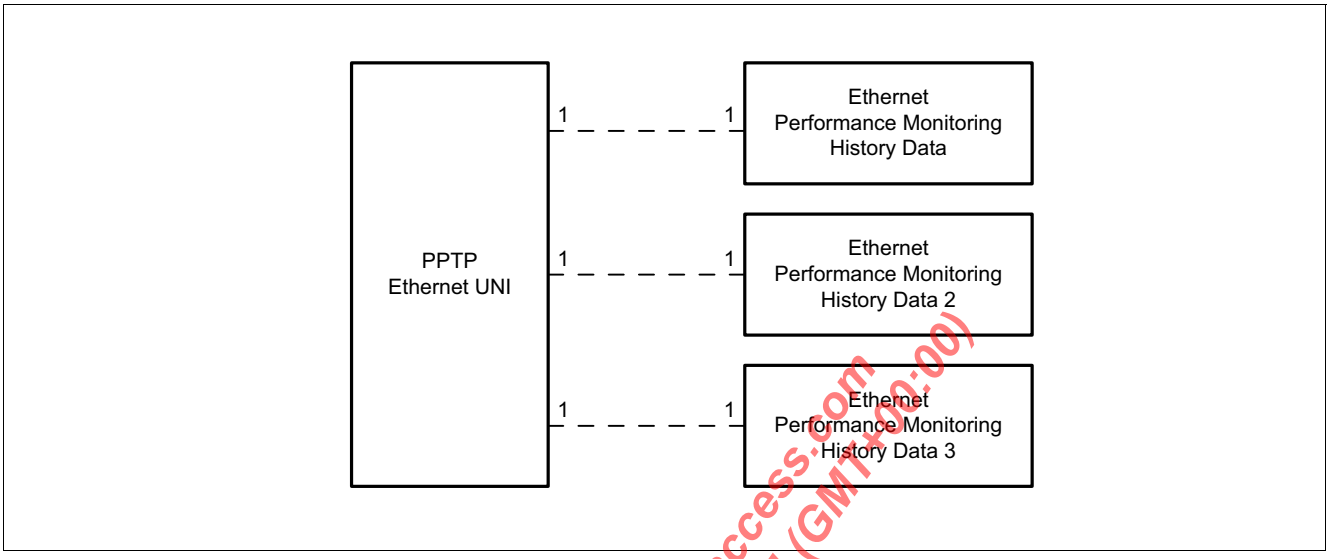


Figure 113 UNI-based Ethernet Performance Monitoring

The hardware counter readout is regularly performed by the PON mailbox driver software (`pon_mbox_drv`).

The readout interval is uncritical (> 1 minute is sufficient), because there is no byte counter included in this set of counters.

The counter values can be read by using the PON adapter function `me_counters_get`.

Managed Entity ID

2 bytes, read-only, set-by-create

This attribute uniquely identifies each instance of this managed entity. Through an identical ID, this managed entity is implicitly linked to an instance of the **PPTP Ethernet UNI** managed entity.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** and **Threshold Data 2** managed entities that are pointed to by this value provide alarm thresholds which are used for the monitoring items.

FCS Errors

4 bytes, read-only

This attribute provides the number of FCS errors that have been detected in upstream (ingress) direction on the selected UNI port.

See the PON adapter data structure `pa_eth_cnt` ().

Excessive Collision Counter

4 bytes, read-only

This attribute provides the number of excessive collision events that have been detected by the Ethernet MAC during half duplex operation.

See the PON adapter data structure `pa_eth_cnt` ().

Late Collision Counter

4 bytes, read-only

This attribute provides the number of late collision events that have been detected by the Ethernet MAC during half duplex operation.

See the PON adapter data structure `pa_eth_cnt` ().

Frames too long

4 bytes, read-only

This attribute provides the number of oversized frame errors that have been detected in upstream (ingress) direction on the selected UNI port.

See the PON adapter data structure `pa_eth_cnt` ().

Buffer Overflow on Receive

4 bytes, read-only

This attribute provides the number of receive buffer overflow events that have been detected.

See the PON adapter data structure `pa_eth_cnt` ().

Buffer Overflow on Transmit

4 bytes, read-only

This attribute provides the number of transmit buffer overflow events that have been detected.

This kind of counter is not supported by the hardware.

See the PON adapter data structure `pa_eth_cnt` ().

Single Collision Frame Counter

4 bytes, read-only

This attribute provides the number of single collision events that have been detected by the Ethernet MAC during half duplex operation.

See the PON adapter data structure `pa_eth_cnt` ().

Multiple Collision Frame Counter

4 bytes, read-only

This attribute provides the number of multiple collision events that have been detected by the Ethernet MAC during half duplex operation.

See the PON adapter data structure `pa_eth_cnt` ().

SQE Counter

4 bytes, read-only

This attribute provides the number of SQE test messages.

See the PON adapter data structure `pa_eth_cnt` ().

Deferred Transmission Counter

4 bytes, read-only

This attribute provides the number of deferred transmission events that have been detected by the Ethernet MAC during half duplex operation.

See the PON adapter data structure `pa_eth_cnt` ().

Internal MAC Transmit Error Counter

4 bytes, read-only

This counter is not supported by hardware. A count value is 0 is returned.

See the PON adapter data structure `pa_eth_cnt` ().

Carrier Sense Error Counter

4 bytes, read-only

This attribute provides the number of carrier sense error events that have been detected by the Ethernet MAC.

See the PON adapter data structure `pa_eth_cnt` ().

Alignment Error Counter

4 bytes, read-only

This attribute provides the number of alignment errors that have been detected by the Ethernet MAC.

See the PON adapter data structure `pa_eth_cnt` ().

Internal MAC Receive Error Counter

4 bytes, read-only

This attribute provides the number of MAC receive errors that have been detected by the Ethernet MAC.

See the PON adapter data structure `pa_eth_cnt` ().

Actions

Create, delete, get, set, get current data

TCA Notifications

TCA #0: FCS errors

TCA #1: Excessive collision counter

TCA #2: Late collision counter

TCA #3: Frames too long counter

TCA #4: Buffer overflow on receive counter

TCA #5: Buffer overflow on transmit counter

TCA #6: Single collision counter

TCA #7: Multiple collisions counter

TCA #8: SQE counter

TCA #9: Deferred transmission counter

TCA #10: Internal MAC transmit error counter

TCA #11: Carrier sense error counter

TCA #12: Alignment error counter

TCA #13: Internal MAC receive error counter

4.20.5.5 Ethernet Performance Monitoring History Data 2

ME class: 89

This managed entity is created by the OLT and provides performance information related to an Ethernet UNI port in upstream direction (from ONU to OLT). See **Figure 113** for the managed entity relationship.

Managed Entity ID

2 bytes, read-only, set-by-create

This attribute uniquely identifies each instance of this managed entity. Through an identical ID, this managed entity is implicitly linked to an instance of the **PPTP Ethernet UNI** managed entity.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** and **Threshold Data 2** managed entities that are pointed to by this value provide alarm thresholds which are used for the monitoring items.

PPPoE Filtered Frame Counter

This attribute provides the number of PPPoE frames that have been dropped upon reception due to the configured filter settings.

See the PON adapter function `eth2_cnt_get`.

Actions

Create, delete, get, set, get current data

TCA Notifications

TCA #0: PPPoE filtered frame counter

4.20.5.6 Ethernet Performance Monitoring History Data 3

ME class: 296

This managed entity is created by the OLT and provides performance information related to an Ethernet UNI port in upstream direction (from ONU to OLT). See **Figure 113** for the managed entity relationship.

The hardware counter readout must be within less than 30 s to avoid an overrun of the counters.

Managed Entity ID

2 bytes, read-only, set-by-create

This attribute uniquely identifies each instance of this managed entity. Through an identical ID, this managed entity is implicitly linked to an instance of the **PPTP Ethernet UNI** managed entity.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** and **Threshold Data 2** managed entities that are pointed to by this value provide alarm thresholds which are used for the monitoring items.

Drop Events

See the PON adapter data structure `pa_eth3_cnt` ().

Octets

The number of received octets (bytes).

See the PON adapter data structure `pa_eth3_cnt` ().

Packets

The number of received packets (received frames)

See the PON adapter data structure `pa_eth3_cnt` ().

Broadcast Packets

The number of received broadcast packets.

See the PON adapter data structure `pa_eth3_cnt` ().

Multicast Packets

The number of received multicast packets.

See the PON adapter data structure `pa_eth3_cnt` ().

Undersized Packets

The number of received undersized packets.

See the PON adapter data structure `pa_eth3_cnt` ().

This count value is always 0.

Fragments

See the PON adapter data structure `pa_eth3_cnt` ().

Jabbers

The number of received jabber (oversized) packets.

See the PON adapter data structure `pa_eth3_cnt` ().

Packets 64 Octets

The number of received 64-byte packets.

See the PON adapter data structure `pa_eth3_cnt` ().

Packets 65 to 127 Octets

The number of received packets with a size between 65 and 127 bytes.

See the PON adapter data structure `pa_eth3_cnt` ().

Packets 128 to 255 Octets

The number of received packets with a size between 128 and 255 bytes.

See the PON adapter data structure `pa_eth3_cnt` ().

Packets 256 to 511 Octets

The number of received packets with a size between 256 and 511 bytes.

See the PON adapter data structure `pa_eth3_cnt` ().

Packets 512 to 1023 Octets

The number of received packets with a size between 512 and 1023 bytes.

See the PON adapter data structure `pa_eth3_cnt` ().

Packets 1024 to 1518 Octets

The number of received packets with a size between 1024 and 1518 bytes.

See the PON adapter data structure `pa_eth3_cnt` ().

Note: An additional counter is provided that counts the number of packets larger than 1518 byte (up to the allowed maximum frame size). This counter is not covered by OMCI.

Actions

Create, delete, get, set, get current data

TCA Notifications

TCA #0: Drop events

TCA #1: Undersized packets

TCA #2: Fragments

TCA #3: Jabbers

4.20.5.7 Ethernet Frame Performance Monitoring History Data Upstream

ME class: 322

This managed entity is created by the OLT and can be attached to ANI-side bridge ports or UNI-side bridge ports. The implementation depends on this location.

4.20.5.7.1 UNI-side Bridge Port PMHD Upstream Implementation

This managed entity provides Ethernet-related performance data for a single UNI-side bridge port in upstream direction. The associated interface (identified by its bridge port index) is defined by the **MAC Bridge Port Configuration Data** managed entity with an identical managed entity ID. The managed entity relationship is illustrated in **Figure 114**.

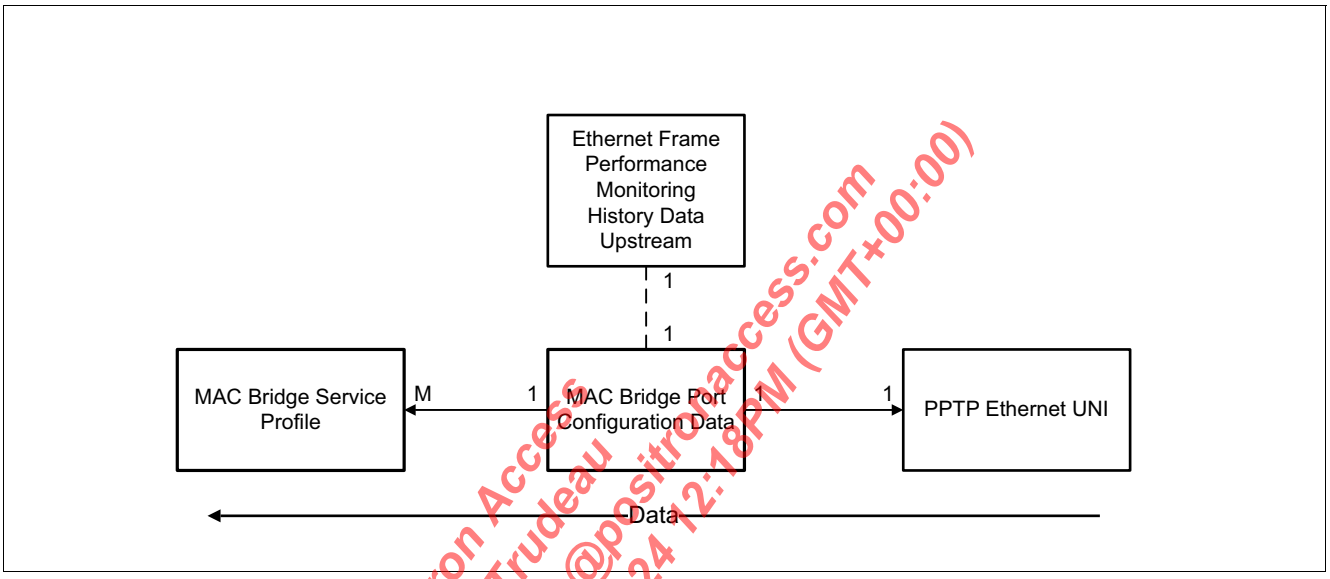


Figure 114 UNI-side Bridge Port Based Ethernet Performance Monitoring Upstream

The availability of this counter set is restricted to bridge ports that are directly assigned to a **PPTP Ethernet UNI** managed entity. Other bridge ports do not provide the required hardware counters and must deliver a constant zero for all counter values. To find the UNI port that is related to a given bridge port, check the **MAC Bridge Port Configuration Data**. If the termination point type is 1 (PPTP Ethernet UNI), follow the termination point pointer to the related **PPTP Ethernet UNI**. The UNI port index is calculated as the lower byte of the managed entity ID minus 1.

Managed Entity ID

2 bytes, read-only, set-by-create

This attribute uniquely identifies each instance of this managed entity. Through an identical ID, this managed entity is implicitly linked to an instance of the **MAC Bridge Port Configuration Data** managed entity.

Interval End Time

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

The **Threshold Data 1** managed entity that is pointed to by this value provides alarm thresholds which are used for the monitoring items. The **Threshold Data 2** managed entity is not used.

Drop Events

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Octets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Broadcast Packets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Multicast Packets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

CRC Errored Packets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Undersized Packets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.
This count value is always 0.

Oversize Packets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 64 Octets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 65 to 127 Octets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 128 to 255 Octets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 256 to 511 Octets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 512 to 1023 Octets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 1024 to 1518 Octets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Note: An additional counter is provided that counts the number of packets larger than 1518 byte (up to the allowed maximum frame size, `lan_cnt_val::ictrll_counter.rx1519`*). This counter is not handled by OMCI.*

4.20.5.7.2 ANI-side Bridge Port PMHD Upstream Implementation

The managed entity provides Ethernet-related performance data for a single ANI-side bridge port in upstream direction (bridge port egress direction). See **Figure 115** for the managed entity relationship.

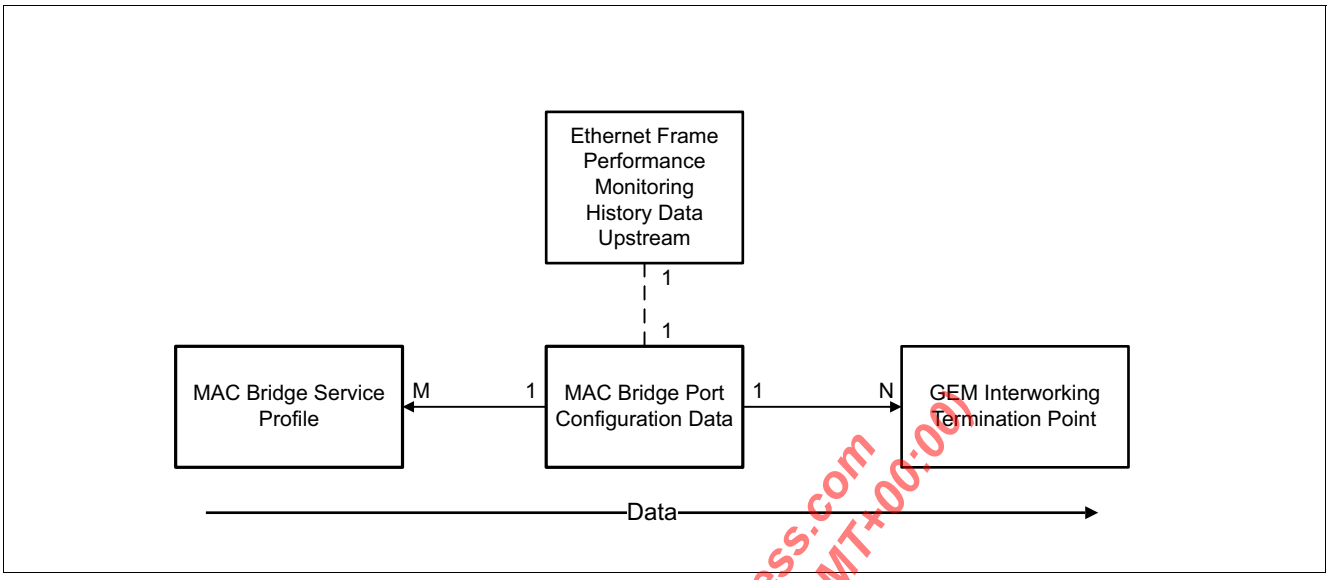


Figure 115 ANI-side Bridge Port Based Ethernet Performance Monitoring Upstream

Managed Entity ID

2 bytes, read-only, set-by-create

This attribute uniquely identifies each instance of this managed entity. Through an identical ID, this managed entity is implicitly linked to an instance of the **MAC Bridge Port Configuration Data** managed entity.

Interval End Time

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

The **Threshold Data 1** managed entity that is pointed to by this value provides alarm thresholds which are used for the monitoring items. The **Threshold Data 2** managed entity is not used.

Drop Events

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Octets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Broadcast Packets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Multicast Packets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

CRC Errored Packets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Undersized Packets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.
This count value is always 0.

Oversize Packets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 64 Octets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 65 to 127 Octets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 128 to 255 Octets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 256 to 511 Octets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 512 to 1023 Octets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 1024 to 1518 Octets

See the PON adapter function `eth_us_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

4.20.5.8 Ethernet Frame Performance Monitoring History Data Downstream

ME class: 321

This managed entity is created by the OLT and can be attached to ANI-side bridge ports or UNI-side bridge ports. The implementation depends on this location.

4.20.5.8.1 UNI-side Bridge Port PMHD Downstream Implementation

This managed entity provides Ethernet-related performance data for a single UNI-side bridge port in downstream direction (bridge port egress direction). See **Figure 116** for the managed entity relationship.

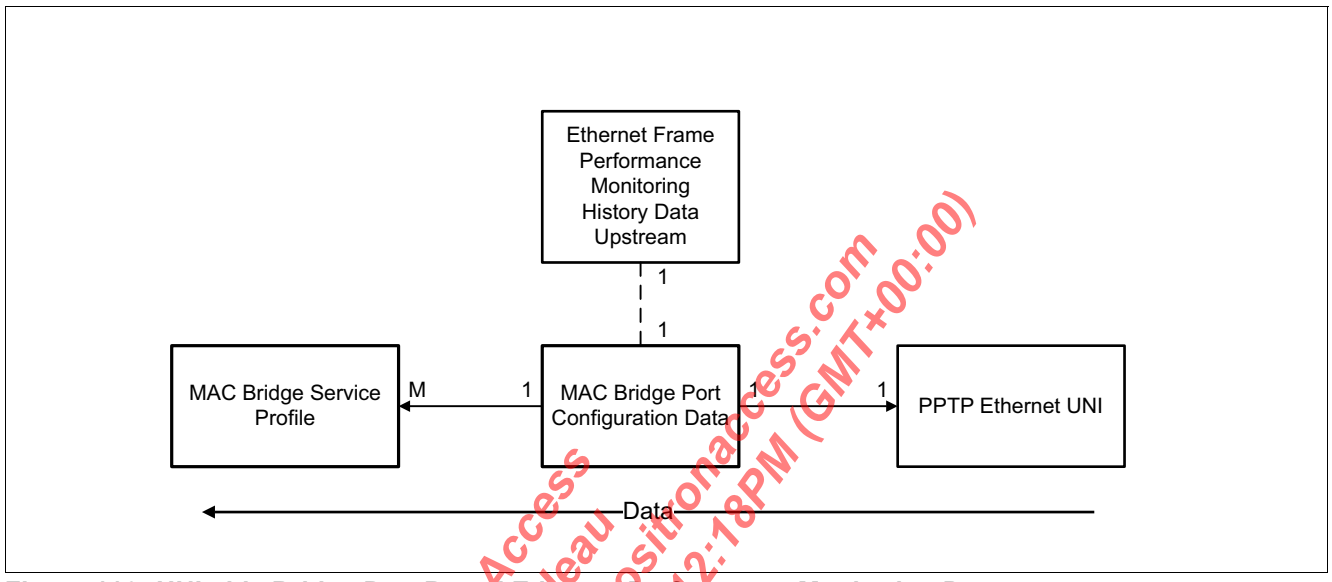


Figure 116 UNI-side Bridge Port Based Ethernet Performance Monitoring Downstream

Managed Entity ID

2 bytes, read-only, set-by-create

This attribute uniquely identifies each instance of this managed entity. Through an identical ID, this managed entity is implicitly linked to an instance of the **MAC Bridge Port Configuration Data** managed entity.

Interval End Time

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

The **Threshold Data 1** managed entity that is pointed to by this value provides alarm thresholds which are used for the monitoring items. The **Threshold Data 2** managed entity is not used.

Drop Events

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Octets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Broadcast Packets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Multicast Packets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

CRC Errored Packets

CRC errored packets are never transmitted. This count value is always 0.

Undersized Packets

Undersized packets are never transmitted. This count value is always 0.

Oversized Packets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 64 Octets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 65 to 127 Octets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 128 to 255 Octets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 256 to 511 Octets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 512 to 1023 Octets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 1024 to 1518 Octets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Note: An additional counter is provided that counts the number of packets larger than 1518 byte (up to the allowed maximum frame size). This counter is not handled by OMCI.

4.20.5.8.2 ANI-side Bridge Port PMHD Downstream Implementation

The managed entity provides Ethernet-related performance data for a single ANI-side bridge port in downstream direction (bridge port ingress direction). See **Figure 114** for the managed entity relationship.

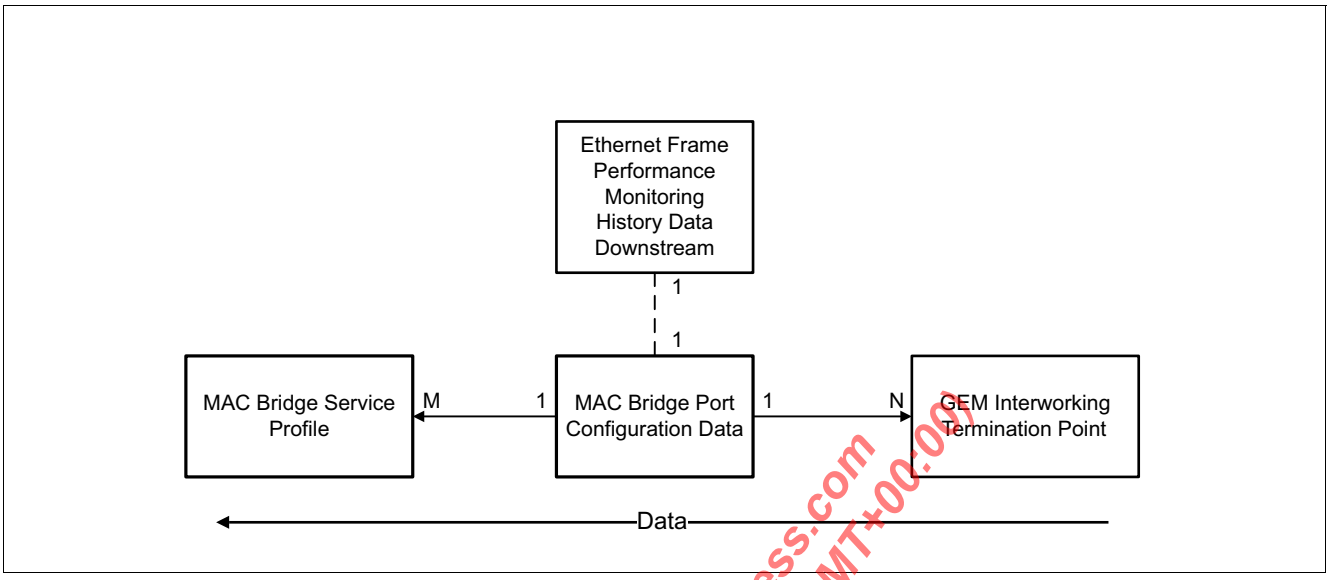


Figure 117 ANI-side Bridge Port Based Ethernet Performance Monitoring Downstream

Managed Entity ID

2 bytes, read-only, set-by-create

This attribute uniquely identifies each instance of this managed entity. Through an identical ID, this managed entity is implicitly linked to an instance of the **MAC Bridge Port Configuration Data** managed entity.

Interval End Time

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

The **Threshold Data 1** managed entity that is pointed to by this value provides alarm thresholds which are used for the monitoring items. The **Threshold Data 2** managed entity is not used.

Drop Events

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Octets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Broadcast Packets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Multicast Packets

Counters per GEM/XGEM port are not provided,

CRC Errored Packets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

This count value is always 0.

Undersized Packets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

This count value is always 0.

Oversize Packets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 64 Octets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 65 to 127 Octets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 128 to 255 Octets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 256 to 511 Octets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 512 to 1023Octets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

Packets 1024 to 1518 Octets

See the PON adapter function `eth_ds_cnt_get` and the data structure `pa_eth_ds_us_cnt`.

4.20.5.9 Ethernet Frame Extended Performance Monitoring History Data

ME class: 334

This is a generic managed entity, created by the OLT, attached to different types of Ethernet termination points within the ONU. Depending on the related parent managed entity class, the selected operational model (SFU, SFP, or HGU), and the available hardware implementation, not all of the counter values are supported. The counters that are not applicable or not supported return a value of 0.

The software implementation depends on the related managed entity attachment point (parent managed entity class), thus multiple cases must be differentiated. An overview of the possible attachment points is given in **Figure 118**. A list of supported counter functions for the various parent managed entity classes is given in **Table 58** and **Table 59**.

As another option, a similar function is provided by the **Ethernet Frame Extended Performance Monitoring History Data 64-bit** managed entity, providing 64-bit counters. The implementation is similar, because the underlying low-level functions always provide 64-bit counter values.

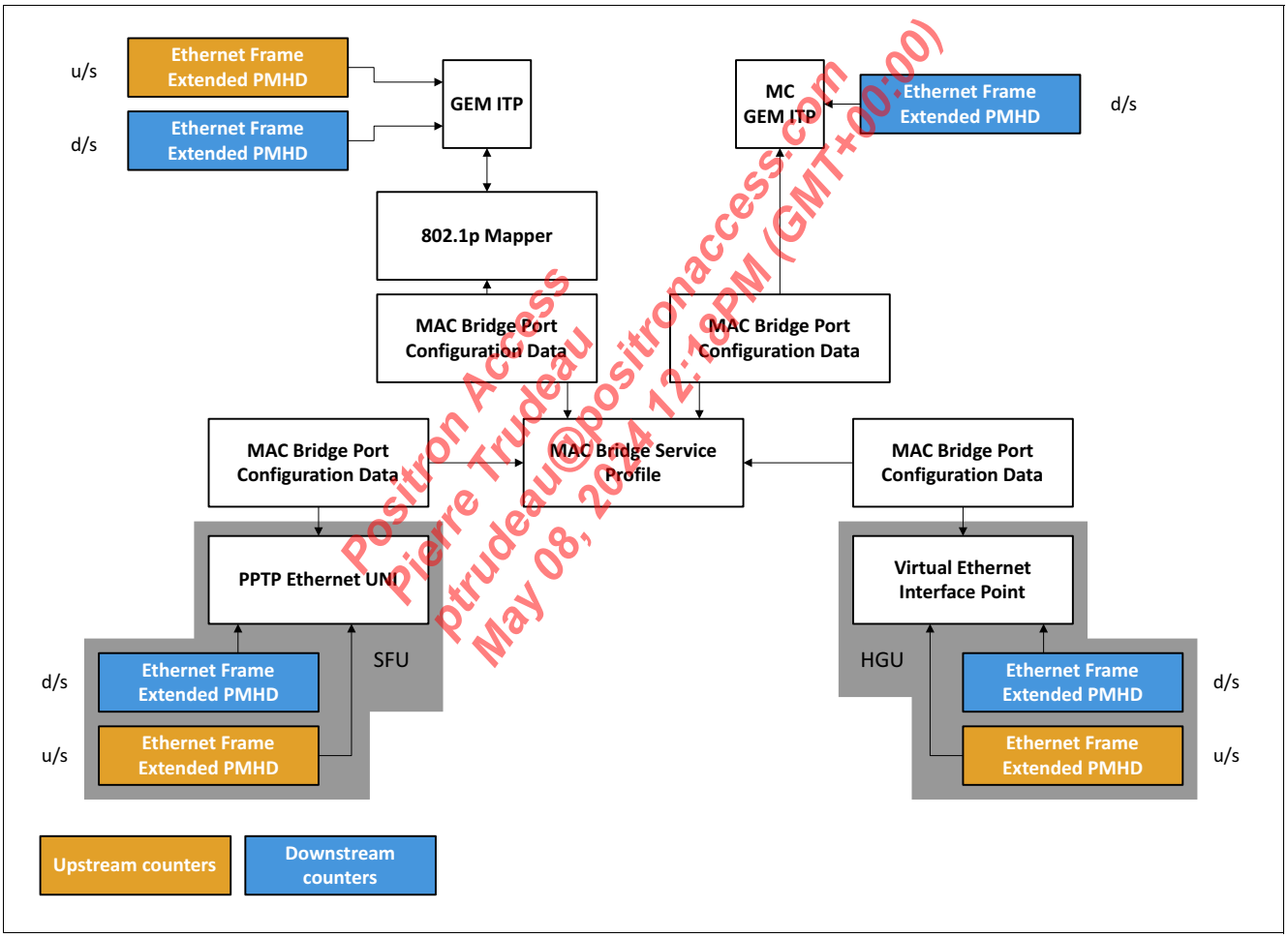


Figure 118 Managed Entity Relationship for Ethernet Frame Extended PMHD

Table 58 Upstream Counter Availability by Parent Managed Entity Type

Parent Managed Entity Type	Upstream Counter Type								
	Drop Events	Octets	Frames	B/C Frames	M/C Frames	CRC Errors	Under-size Frames	Over-size Frames	Frames <n> Octets
PPTP Ethernet UNI	No	Yes	Yes	No	Yes	Yes	No	No	Yes
GEM Interworking Termination Point	N/A	Yes	Yes	No	No	N/A	N/A	N/A	Yes
Virtual Ethernet Interface Point	N/A	Yes	Yes	No	Yes	N/A	N/A	N/A	No

Table 59 Downstream Counter Availability by Parent Managed Entity Type

Parent Managed Entity Type	Downstream Counter Type								
	Drop Events	Octets	Frames	B/C Frames	M/C Frames	CRC Errors	Under-size Frames	Over-size Frames	Frames <n> Octets
PPTP Ethernet UNI	N/A	Yes	Yes	No	Yes	N/A	No	N/A	Yes
GEM Interworking Termination Point	N/A	Yes	Yes	No	Yes	N/A	No	No	Yes
Multicast GEM Interworking Termination Point	N/A	Yes	Yes	No	Yes	N/A	No	No	Yes
Virtual Ethernet Interface Point	N/A	Yes	Yes	No	Yes	N/A	N/A	N/A	No

Managed Entity ID

2 bytes, read-only, set-by-create

This attribute uniquely identifies each instance of this managed entity. The value is selected by the OLT, a subsequent numbering starting at 1 is recommended.

Interval End Time

1 byte, read-only.

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Control Block

16 bytes, read, write, set-by-create.

This attribute contains multiple fields, which are:

- **Control Block: Threshold Data 1/2**
- **Control Block: Parent Managed Entity Instance**
- **Control Block: Accumulation Disable**
- **Control Block: TCA Disable**

- **Control Block: Control Fields**
- **Control Block: TCI**

A detailed description of the control block fields is provided hereafter.

Control Block: Threshold Data 1/2

2 bytes, read, write, set-by-create.

The **Threshold Data 1** managed entity pointed to by this value provides alarm thresholds used for the monitoring items. The **Threshold Data 2** managed entity is not used.

Control Block: Parent Managed Entity Class

2 bytes, read, write, set-by-create.

This field contains the enumerated value of the ME class of the performance monitoring managed entity parent. Together with the parent managed entity instance field, this permits a given performance monitoring managed entity to be associated with any OMCI managed entity of that type. The supported managed entity classes are:

- 11: **PPTP Ethernet UNI**
- 266: **GEM Interworking Termination Point**
- 281: **Multicast GEM Interworking Termination Point**
- 329: **Virtual Ethernet Interface Point**

Control Block: Parent Managed Entity Instance

2 bytes, read, write, set-by-create.

This field identifies the specific parent managed entity instance to which this performance monitoring is attached by its managed entity ID value.

If no match is found for the parent managed entity class and the parent management ID in the MIB, an error response is returned to the OLT.

Control Block: Accumulation Disable

2 bytes, read, write, set-by-create.

This bit field allows performance monitoring accumulation to be disabled. The default value 0 enables performance monitoring data collection. Each bit set in this attribute disables the related counter. See **Table 60** (bit positions are numbered 0 to 13, related to counter types 1 to 14).

Bit 14 is used to disable all counters. When set to 1, the configuration of bit position 0 to 13 is ignored.

Bit 15 is used to trigger a clearing of all counters. Individual counters cannot be cleared.

Table 60 Bit Assignment for Accumulation

Bit	15	14	13	…	1	0
Accumulation Disable	Global clear	Global disable	PM14		PM2	PM1

Control Block: TCA Disable

2 bytes, read, write, set-by-create.

This field permits threshold crossing alerts to be inhibited, either individually or for the complete managed entity instance. Each bit set in this attribute disables the related threshold. Only bit positions 0 to 3 are used by OMCI (for TCA types 1 to 4) and only bit position 1 (TCA #2) is supported by the software. See **Table 61**.

Table 61 Bit Assignment for TCA Disable

Bit	15	14	13	…	2	1	0
TCA Disable	Unused (ignored)	Global disable	Unused (ignored)		Unused (ignored)	CRC Errored Frames	Unused (ignored)

Control Block: Control Fields

2 bytes, read, write, set-by-create.

This field is a bit map whose values govern the behavior of this performance monitoring managed entity (control fields are numbered 1 to 16, related to bit positions 0 to 15).

Table 62 Control Field Usage

Bit	Usage
1 (LSB)	The value 1 specifies continuous accumulation, regardless of 15-minute intervals. There is no concept of current or historic accumulators; get and get current data (when supported) both return current values. The value 0 specifies 15-minute accumulators exactly like those of classical PM.
2[1)]	This bit indicates the direction for the data collection. The value 0 indicates that data is to be collected for the upstream traffic. The value 1 indicates that data is to be collected for the downstream traffic.
3 … 14	Reserved, must be set to 0 by the OLT and must be ignored by the ONU.
15[2)]	When this bit is 1, the priority bits (p-bits) of the TCI field are used to filter the PM data collected. The value 0 indicates that PM is collected without regard to p-bits.
16[3)]	When this bit is 1, the VID bits of the TCI field are used to filter the PM data collected. The value 0 indicates that PM is collected without regard to VID.

1) Upstream counters are not supported when connected to a multicast GEM port.
2) Filtering by p-bits is not supported when connected to a unicast or multicast GEM port.
3) Filtering by VID is not supported when connected to a unicast or multicast GEM port.

Control Block: TCI

2 bytes, read, write, set-by-create.

This field contains the value optionally used as a filter for the performance monitoring data collected, under the control of bits 14 and 15 of the control fields. These rules apply:

This value is matched to the outer tag of a frame. The untagged frames are not counted when this field is used.

- This value is matched to the outer tag of a frame, when the frame is double-tagged.
- The untagged frames are not counted when this field is used.
- The DEI bit (bit 12 of the TCI) is always ignored and cannot be configured for filtering.

General Counter Handling

The counters are provided by the `pon_adapter` layer (see `pon_adapter_ethernet_pmhd.h`):

```
/** Retrieve current values of Ethernet Extended PMHD total counters
*
* \param[in]  ll_handle              Lower layer context pointer
* \param[in]  parent_class_id        Parent Managed Entity class
*                                    identifier
* \param[in]  parent_me_id           Parent Managed Entity instance
*                                    identifier
```

```
* \param[in]  dir                      Counters direction
* \param[out] eth_ext_cnt              Counters values
*/
enum pon_adapter_errno (*eth_ext_cnt_get)(
       void *ll_handle,
       const uint16_t parent_class_id,
       const uint16_t parent_me_id,
       const enum pon_adapter_eth_ext_pmhd_dir dir,
       struct pa_eth_ext_cnt *eth_ext_cnt);
```

This function allows to specify the related managed entity (`class_id` and `instance_id`) and the traffic direction (`PON_ADAPTER_ETH_EXT_PMHD_DIR_US` or `PON_ADAPTER_ETH_EXT_PMHD_DIR_DS`).
The counter values are held in a common data structure (`struct pa_eth_ext_cnt`).

Drop Events

4 bytes, read-only

Read `drop_events` of `pa_eth_ext_cnt`.

Octets

4 bytes, read-only

Read `octets` of `pa_eth_ext_cnt`.

Frames

4 bytes, read-only

Read `frames` of `pa_eth_ext_cnt`.

Broadcast Frames

The total number of good frames directed to the broadcast address. This does not include multicast frames.

4 bytes, read-only

Read `bc_frames` of `pa_eth_ext_cnt`.

Multicast Frames

The total number of good frames directed to a multicast address. This does not include broadcast frames.

Read `mc_frames` of `pa_eth_ext_cnt`.

CRC Errored Frames

4 bytes, read-only

Read `crc_err_frames` of `pa_eth_ext_cnt`.

Undersized Frames

4 bytes, read-only

Read `usize_frames` of `pa_eth_ext_cnt`.

This count value is always 0.

Oversized Frames

4 bytes, read-only

Read `osize_frames` of `pa_eth_ext_cnt`.

Frames 64 Octets

4 bytes, read-only

Read `frames_64` of `pa_eth_ext_cnt`.

Frames 65 to 127 Octets

4 bytes, read-only

Read `frames_65_127` of `pa_eth_ext_cnt`.

Frames 128 to 255 Octets

4 bytes, read-only

Read `frames_128_255` of `pa_eth_ext_cnt`.

Frames 256 to 511 Octets

4 bytes, read-only

Read `frames_256_511` of `pa_eth_ext_cnt`.

Frames 512 to 1023 Octets

4 bytes, read-only

Read `frames_512_1023` of `pa_eth_ext_cnt`.

Frames 1024 to 1518 Octets

4 bytes, read-only

Read `frames_1024_1518` of `pa_eth_ext_cnt`.

Actions

Create, delete, get, set, get current data

TCA Notifications

TCA #1: Drop events counter

TCA #2: CRC errored frames counter

TCA #3: Undersized frames counter

TCA #4: Oversized frames counter

4.20.5.10 Ethernet Frame Extended Performance Monitoring History Data 64-bit

ME class: 425

This is a generic managed entity, created by the OLT, that can be attached to different types of Ethernet termination points within the ONU. Depending on the related managed entity, the selected operational model (SFU, MDU, or HGU), and the available hardware implementation, not all of the counter values might be supported. Counters that are not applicable or not supported return a value of 0.

The software implementation depends on the related managed entity, thus multiple cases must be differentiated. An overview of the possible attachment points is given in **Figure 119**. A list of supported counter functions for the various parent managed entity types is given in **Table 58** and **Table 59**.

In contrast to the **Ethernet Frame Extended Performance Monitoring History Data** managed entity, this one can accumulate counter values up to 64 bit and thus does not need to be checked within a 15-minute interval for counter accumulation in the OLT.

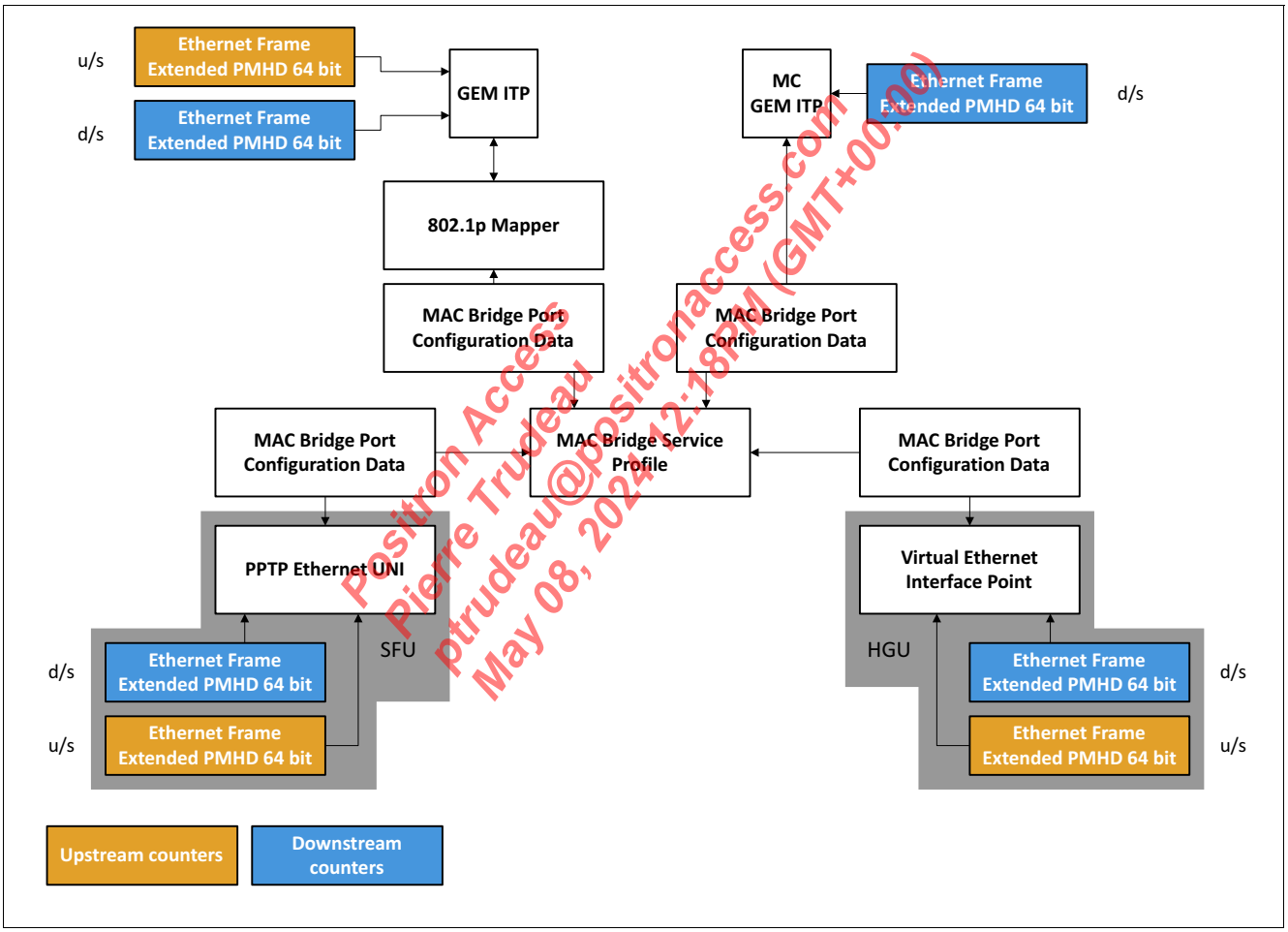


Figure 119 Managed Entity Relationship for Ethernet Frame Extended PMHD

Managed Entity ID

2 bytes, read-only, set-by-create

This attribute uniquely identifies each instance of this managed entity. The value is selected by the OLT, a subsequent numbering starting at 1 is recommended.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Control Block

16 bytes, read, write, set-by-create.

This attribute contains multiple fields, which are:

- **Control Block: Threshold Data 64-bit**
- **Control Block: Parent Managed Entity Class**
- **Control Block: Accumulation Disable**
- **Control Block: TCA Disable**
- **Control Block: Control Fields**
- **Control Block: TCI**

See below for a detailed description of the control block fields.

Control Block: Threshold Data 64-bit

2 bytes, read, write, set-by-create.

The **Threshold Data 64-bit** managed entity that is pointed to by this value provides alarm thresholds which are used for the monitoring items.

Control Block: Parent Managed Entity Class

2 bytes, read, write, set-by-create.

This field contains the enumerated value of the ME class of the Performance Monitoring managed entity’s parent. Together with the parent managed entity instance field, this permits a given Performance Monitoring managed entity to be associated with any OMCI managed entity of that certain type. The supported managed entity classes are:

- 11: **PPTP Ethernet UNI**
- 266: **GEM Interworking Termination Point**
- 281: **Multicast GEM Interworking Termination Point**
- 329: **Virtual Ethernet Interface Point**

Control Block: Parent Managed Entity Instance

2 bytes, read, write, set-by-create.

This field identifies the specific parent managed entity instance to which this Performance Monitoring is attached by its managed entity ID value.

Control Block: Accumulation Disable

2 bytes, read, write, set-by-create.

This bit field allows PM accumulation to be disabled. The default value 0 enables performance monitoring data collection. Each bit that is set in this attribute disables the related counter. See **Table 63** (bit positions are numbered 0 to 15).

Table 63 Bit Assignment for Accumulation/TCA Disable

Bit	15	14	13	…	1	0
Accumulation Disable	Global clear	Global disable	PM14		PM2	PM1
TCA Disable		Global disable	TH14		TH2	TH1

Control Block: TCA Disable

2 bytes, read, write, set-by-create.

This field permits TCAs to be inhibited, either individually or for the complete managed entity instance. Each bit that is set in this attribute disables the related threshold. See **Table 63**.

Control Block: Control Fields

2 bytes, read, write, set-by-create.

This field is a bit map whose values govern the behavior of this Performance Monitoring managed entity (bit positions are numbered 0 to 15). See **Table 64** for the bit definitions.

Table 64 Control Field Usage

Bit	Usage
1 (LSB)	The value 1 specifies continuous accumulation, regardless of 15-minute intervals. There is no concept of current and historic accumulators; get and get current data (if supported) both return current values. The value 0 specifies 15-minute accumulators exactly like those of classical PM.
2	This bit indicates the direction for the collection of data. The value 0 indicates that data is to be collected for upstream traffic. The value 1 indicates that data is to be collected for downstream traffic.
3 … 14	Reserved, should be set to 0 by the OLT and ignored by the ONU.
15[1)]	When this bit is 1, the priority bits (p-bits) of the TCI field are used to filter the PM data collected. The value 0 indicates that PM is collected without regard to p-bits.
16[2)]	When this bit is 1, the VID bits of the TCI field are used to filter the PM data collected. The value 0 indicates that PM is collected without regard to VID.

1) Filtering by p-bits is not supported.
2) Filtering by VID is not supported.

Control Block: TCI

2 bytes, read, write, set-by-create.

This field contains the value optionally used as a filter for the Performance Monitoring data collected, under the control of bits 14 and 15 of the control fields. This value is matched to the outer tag of a frame. Untagged frames are not counted when this field is used.

Note: Filtering by VLAN ID and/or priority is not supported by the underlying hardware/firmware counter implementation. The TCI value is ignored.

Counter Implementation

The implementation of each counter depends on the parent managed entity and data direction (upstream or downstream) that is monitored. Not all counters are available for each type of supervised managed entity or data direction.

For implementation details, please follow the description of the **General Counter Handling** in the **Ethernet Frame Extended Performance Monitoring History Data** managed entity section. The underlying software layers provide 64-bit counter structures and can be used in the same way.

Actions

Create, delete, get, set, get current data

TCA Notifications

TCA #1: Drop events counter

TCA #2: CRC errored frames counter

TCA #3: Undersized frames counter

TCA #4: Oversized frames counter

4.20.6 General Purpose Managed Entities

These managed entities are defined by ITU-T G.988 **[56]** in chapter 9.12.

4.20.6.1 UNI-G

ME class: 264

This managed entity is created by the ONU autonomously upon OMCI startup. The hardware programming to implement this managed entity depends on the related user network interface. The related interface is identified by an identical management entity ID and is either an Ethernet PPTP UNI a PPTP xDSL UNI, or a POTS PPTP UNI.

The reference implementation covers only the PPTP Ethernet UNI.

Managed Entity ID

2 bytes, read-only

This attribute uniquely identifies each instance of this managed entity. Through an identical ID, this managed entity is implicitly linked to an instance of an Ethernet PPTP UNI.

Configuration Option Status

This attribute is deprecated and must not be used. It should be set to 0 by the OLT and ignored by the ONU.

Administrative State

1 byte, read, write

This attribute locks (disables) or unlocks (enables) the function of the related interface.

- 0: The interface is unlocked and is able to transmit data.
- 1: The interface is locked, no data transmission is possible.

The implementation of this attribute depends on the related interface type.

Note: The related PPTP UNI managed entities both provide an own administrative state attribute. An interface must be enabled only when both of them are set to unlocked.

Note: This rule may be violated by some OLTs, in this case enable the interface based on the OLT requirements.

Management Capability

1 byte, read-only

This attribute indicates which management protocol is used to control the related interface.

- 0: OMCI only
- 1: Non-OMCI only
- 2: OMCI or non-OMCI

Note: Only the attribute value of 0 is supported by the reference implementation.

Non-OMCI Management Identifier

2 bytes, read, write

This parameter informs the OLT whether the associated PPTP is solely managed by OMCI or by another management protocol handled through a virtual Ethernet interface point. The possible relationships are:

- **PPTP Ethernet UNI**
- **PPTP POTS UNI**
- **PPTP xDSL UNI Part 1**

This attribute is set to 0, when OMCI is the management protocol. It is set to a pointer to a **Virtual Ethernet Interface Point**, when a different management protocol is used.

Note: Only the attribute value of 0 is supported by the reference implementation.

Relay Agent Options

2 bytes, read, write

This attribute is a pointer to a **Large String** managed entity whose contents specifies one or more DHCP relay agent options.

Any implementation beyond the reference is application specific and must be added by the system vendor.

Actions

Get, set

Notifications

This managed entity does not generate notifications.

4.20.6.2 OLT-G

ME class: 131

This managed entity is created autonomously by the ONU, used to identify the OLT to the ONU and to enable the ITU PON-specific time of day evaluation.

Manage Entity ID

2 bytes, read-only

This attribute identifies the instance of this managed entity. There is only one instance, the value is 0.

OLT Vendor ID

4 bytes, read, write.

This attribute identifies the OLT vendor.

If the ONU is provided by the same vendor, this attribute typically reflects the four most significant bytes used to create the ONU serial number. Upon instantiation, this value is filled with four spaces.

Well-known identifiers are (in alphabetical order):

- ADTN: Adtran
- ALCL: Nokia (was Alcatel-Lucent before)
- CIEN: Ciena
- DZSI: Dasan/Zhone (DZS)
- HWTC: Huawei
- TBIT: Ciena (was Tibit before)

The received string can be used to implement an ONU behavior that depends on a specific OLT vendor.

The PON library function `fapi_pon_olt_type_set` is used to apply such handling.

```
# pon olt_type_set
Long Form: olt_type_set
Short Form: ots

Input Parameter
- enum olt_type type
    PON_OLT_UNKNOWN = 0
    PON_OLT_ADTRAN = 1
    PON_OLT_CALIX = 2
    PON_OLT_CIENA = 3
    PON_OLT_DZS = 4
    PON_OLT_HUAWEI = 5
    PON_OLT_NOKIA = 6
    PON_OLT_TIBIT = 7
    PON_OLT_ZTE = 8

Output Parameter
- enum fapi_pon_errorcode errorcode
```


Equipment ID

20 bytes, read, write.

This 20-byte attribute is used to identify a specific OLT type. By default, it is filled with all spaces.

The usage of this value must be added by the system vendor.

Version

14 bytes, read, write.

This is a 14-byte attribute to define the OLT version. When no version is available, the string is set to a left-justified string of 0 with trailing NULL characters.

The usage of this value must be added by the system vendor.

Time of Day Information

14 bytes, read, write.

Hardware programming is required to implement the time-of-day (ToD) function. This function receives the exact time related to the **TAI** time scale (international atomic time) from the ITU PON OLT and transmits the time related to the **GPS** time scale on the serial interface.

Tho enable the ToD function is the system, the following software packages must be enabled:

- `ptp4l`
- `tod2phc`

See **Chapter 9.2**, **PON Time of Day** for more detail.

Actions

Get, set.

ToD Byte Handling

Upon reception of this attribute, the new time of day is set by using `fapi_pon_gpon_tod_sync_set` with:

- `multiframe_count` = byte 0 to 3 of the 14-byte attribute, the lower 30 bit are used.
- `tod_extended_seconds` = byte 4 to 5 of the 14-byte attribute.
  - Byte 4 and 5 are expected to be zero until beginning of the year 2106.
- `tod_seconds` = byte 6 to 9 of the 14-byte attribute, the lower 31 bit are used.
- `tod_nano_seconds` = byte 10 to 13 of the 14-byte attribute.

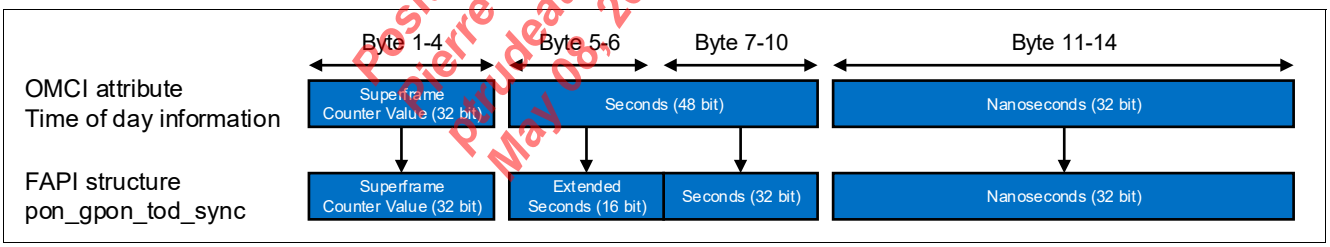


Figure 120 ToD Byte Handling

Due to a variable receive path delay, a compensation is required. The compensation time `t_corr` in units of one second is calculated as:

```
t_corr = (31 - gtc_ds_delay) / 2.48832e9;
```

The hardware delay changes each time the receive path resynchronizes to the optical signal after a loss of signal condition. This correction is included in the `onu_resp_time` value read through `fapi_pon_gpon_status_get` and is read each time a new time of day attribute value is received from the OLT.

The resulting current time of day is read by using `fapi_pon_tod_get`.

Note: There is a known fiber length dependency of the time accuracy, caused by the different light speed values for upstream and downstream (1310 nm and 1490 nm) in a given fiber plant. The value is adapted by modifying `GTC_GROUP_VELOCITY_INDEX`.

Note: There is another inaccuracy of about 26 ns (32 upstream bit times), which cannot be compensated. It changes with each CDR resynchronization.

In addition to the ranging delay time, the preassigned delay must be taken into account to cover a fiber length of more than 20 km.
For details, see `tod_reload_fetch in drv_onu_ll_tod.c`.

1PPS Pulse Output

A dedicated output is provided as a secondary function of a GPIO signal to transmit a synchronization signal as described in **Figure 121**.

The GPIO interface to be used is defined in the device tree (see `&pinctr, pin_1pps_disabled`).

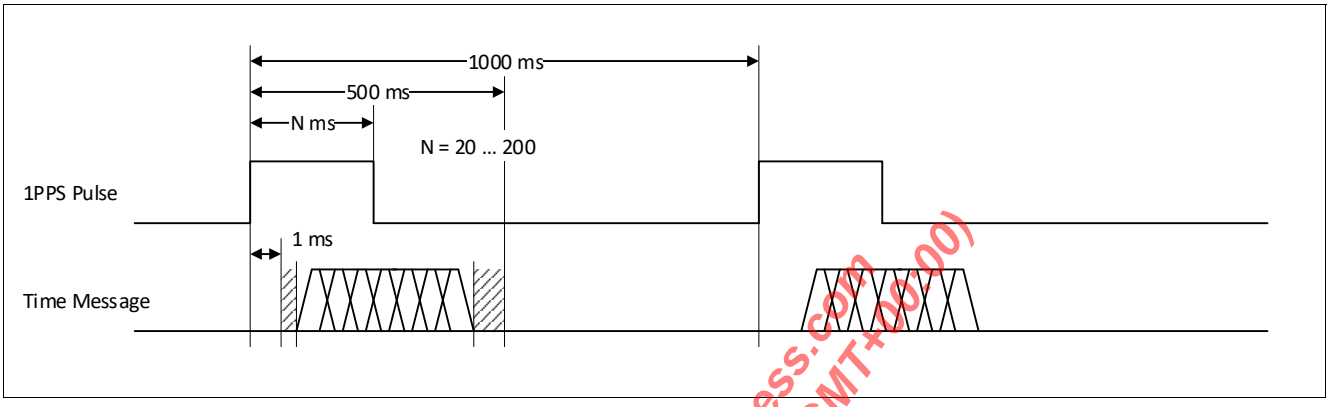


Figure 121 1PPS Signal Timing Characteristics

The pulse width is configurable in the `fapi_pon_tod_init` function (`pps_pulse_width`), in steps of 100 µs. The maximum pulse width is N = 409.5 ms.

1PPS Interrupt Routine

The generation of a 1PPS (one pulse per second) signal is outside the scope of OMCI configuration. It is described here due to its tight relationship to the OMCI time synchronization method.

An interrupt routine is required to generate a serial output once per second. The interrupt is triggered by the time of day hardware each time a rising edge of the 1PPS pulse is generated. The current time is latched by hardware and is read through `fapi_pon_tod_get`. From this value, the GPS time format is computed. The resulting value plus one second is sent through the serial interface (ASC). The time value represents the time valid with the current 1PPS rising edge.

The latched time data format is given in number of seconds, referenced to the TAI time scale (counting started at 1970-01-01 00:00).

The time format sent in the serial protocol is the GPS format, providing a number of weeks since 1980-01-06 00:00 and the number of seconds from beginning of the current week.

The GPS time is synchronized with UTC and TAI time but not adjusted for leap seconds. **Table 65** shows there is an integer number of seconds of offset from each.

Table 65 Time System Relationship

	From			
To		TAI	UTC	GPS
	TAI	0	+37[1)]	+19
	UTC	-37[1)]	0	+18[1)]
	GPS	-19	+18[1)]	0

1) Variable, leap seconds as of February 2024. The last leap second has been inserted on the transition from 2016-12-31 to 2017-01-01, increasing the difference between UTC and TAI to 37 s and between UTC and GPS to 18 s.

When the OLT uses TAI as a time reference (as used for IEEE 1588 PTP timing systems), the variable number of leap seconds is of no significance. To calculate the UTC time, the number of leap seconds must be accounted for. For updates on the leap seconds, refer to **https://hpiers.obspm.fr/eop-pc/index.php** and **https://hpiers.obspm.fr/iers/bul/bulc/Leap_Second.dat**.

The automatic leap second handling for UTC calculation is outside the scope of the OMCI software delivery and must be covered by using the appropriate Linux functionality.

The actual time related with a 1PPS pulse is defined by a serial data transmission on the ToD serial interface (RS-232 compliant output). Each message is related to the most recent 1PPS rising edge that was present immediately before the start of the message. The serial transmission must start latest 1 ms after the rising edge of the 1PPS pulse was sent but not earlier and must be completed within 500 ms after the rising edge, which makes up half of the 1PPS pulse period.

The 1PPS pulse shape and an optional interrupt delay (typically 1 ms) are configured by using the `fapi_pon_tod_init` function (`pps_pulse_width`, `interrupt_delay`).

The data rate of the serial transmission is defined by the application and configurable through the ASC driver functions. The default configuration is:

- 9600 bd
- No parity
- 8 bit data
- One stop bit

These are the default settings used for serial interface on the reference and evaluation boards. Depending on the target application, these parameters may be adapted.

Time of Day Serial Protocol – ITU-T G.8271

Note: The serial output provided by the 10G PON Chipset System Package is just an implementation example. It must be adapted to the specific application requirements by the ONU vendor, if required.

This time format is also known as V.11 interface. The time and date information delivered as a serial ASCII data stream. The G.8271 protocol has been defined by the ITU. Full support requires an extra 1PPS and ToD input to compensate for any variable wire latency.

A single message is sent once per second. **Table 66** shows its composition. The message length is variable, depending on the message contents.

There are three types of message contents defined:

- Time event message
  - Timestamp and basic traceability information
    This message is transmitted by all clock types using this interface.
- Time announce message
  - Virtual PTP announce message.
    This message is transmitted by a PTP clock.
- GNSS status message
  - Provides information about the status of a GNSS timing receiver.
    This message is transmitted by a GNSS-based clock.

For details regarding the message contents, refer to chapter A.1.3 of the ITU-T G.8271/Y.1366 **[60]** latest revision.

Table 66 Time of Day Serial Protocol – G.8271

Byte Number	Contents	Note
Header Part		
1	Synchronization character 1 SYNC CHAR 1	Fixed value of 43_H (“C”)

Table 66 Time of Day Serial Protocol – G.8271 (cont'd)

Byte Number	Contents	Note
2	Synchronization character 2 SYNC CHAR 2	Fixed value of $4D_H$ ("M")
3	Header CLASS	Message class identifier
4	Header ID	Message subtype identifier
5	Length	Message payload length, not including synchronization bytes, header bytes, length, and FCS.
6		
7 … N	Payload	Message payload of variable size, as given by the length parameter.
N+1	Frame Checksum FCS	The checksum includes message header, length, payload and FCS field. The polynomial for generating the FCS is: G(x) = x8 + x5 + x4 + 1. The initial value of the field must be set to 0xFF. The right-shift operation must be used. There must be no bit inversion of the input and output value. The transmission of the octet follows the same bit order as other data octets.

Time of Day Serial Protocol – NMEA

Note: The serial output provided by the 10G PON Chipset System Package is just an implementation example. It must be adapted to the specific application requirements by the ONU vendor, when required.

The time and date information is delivered as a serial ASCII data stream. The NMEA 0183 protocol covers a large number of applications.

A single message is sent once per second, representing the GPS time (`$GPZDG`) or UTC time (`$GPZDA`) output of a standard GPS receiver, such as:

```
$GPZDG,204237.0,02,05,2013,00.00,2*7B
```

or

```
$GPZDA,204237.00,02,05,2013,00,00*61
```

The serial output is simulated without OLT connection by using the free-running test mode selected by `fapi_pon_tod_init` with `tod_debug = 1`.

See **Table 67** for details. An example implementation of the interrupt handler is provided in the ONU driver, see the functions `tod_asc_handle` in `drv_onu_tod.c` and `drv_onu_tod_asc.c`.

Table 67 Time of Day Serial Protocol – NMEA

Byte Number	Contents	Note
$GPZDG Sentence		
1	Start string for GPS time Byte 1: $ Byte 2-3: Talker identifier Byte 4-5: Sentence identifier	Fixed value of 24_H ("$")
2		Fixed value of 47_H ("G")
3		Fixed value of $50x_H$ ("P")
4		Fixed value of $5A_H$ ("Z")
5		Fixed value of 44_H ("D")
6		Fixed value of 47_H ("G")
7	Separator	Fixed value of $2C_H$ (",")

Table 67 Time of Day Serial Protocol – NMEA (cont'd)

Byte Number	Contents	Note
8	Hours	Two-byte variable value from "00" to "23"
9		
10	Minutes	Two-byte variable value from "00" to "59"
11		
12	Seconds	Two-byte variable value from "00" to "59"
13		
14	Decimal point	Fixed value of $2E_H$ (".")
15	1/10 seconds	Fixed value of 30_H ("0")
16	Separator	Fixed value of $2C_H$ (",")
17	Day	Two-byte variable value from "01" to "31"
18		
19	Separator	Fixed value of $2C_H$ (",")
20	Month	Two-byte variable value from "01" to "12"
21		
22	Separator	Fixed value of $2C_H$ (",")
23	Year	One-byte variable value from "0" to "9".
24		One-byte variable value from "0" to "9".
25		One-byte variable value from "0" to "9".
26		One-byte variable value from "0" to "9".
27	Separator	Fixed value of $2C_H$ (",")
28	Signal strength	Fixed dummy value of 30_H ("0")
29		Fixed dummy value of 30_H ("0")
30	Decimal point	Fixed value of $2E_H$ (".")
31	Signal strength	Fixed dummy value of 30_H ("0")
32		Fixed dummy value of 30_H ("0")
33	Separator	Fixed value of $2C_H$ (",")
34	Synchronization status	Value of 30_H ("0") while waiting for synchronization or the optical connection is lost, value of 32_H ("2") while in sync with OLT. • 0: Invalid time • 1: +/- 20 ms • 2: +/- 100 ns
35	Checksum separator	Fixed value of $2A_H$ ("*")
36	Checksum	XOR all bytes between "$" and "*", not including these characters.
37		*Note: Must be calculated whenever the frame contents changes.*
38	Message termination	Fixed value of $0D_H$ ([CR])
39		Fixed value of $0A_H$ ([LF])
$GPZDA Sentence		

Table 67 Time of Day Serial Protocol – NMEA (cont'd)

Byte Number	Contents	Note
1	Start string for UTC time Byte 1: $ Byte 2-3: Talker identifier Byte 4-5: Sentence identifier	Fixed value of 24_H ("$")
2		Fixed value of 47_H ("G")
3		Fixed value of $50x_H$ ("P")
4		Fixed value of $5A_H$ ("Z")
5		Fixed value of 44_H ("D")
6		Fixed value of 40_H ("A")
7	Separator	Fixed value of $2C_H$ (",")
8	Hours	Two-byte variable value from "00" to "23"
9		
10	Minutes	Two-byte variable value from "00" to "59"
11		
12	Seconds	Two-byte variable value from "00" to "59"
13		
14	Decimal point	Fixed value of $2E_H$ (".")
15	1/10 seconds	Fixed value of 30_H ("0")
16	1/100 seconds	Fixed value of 30_H ("0")
17	Separator	Fixed value of $2C_H$ (",")
18	Day	Two-byte variable value from "01" to "31"
19		
20	Separator	Fixed value of $2C_H$ (",")
21	Month	Two-byte variable value from "01" to "12"
22		
23	Separator	Fixed value of $2C_H$ (",")
24	Year	One-byte variable value from "0" to "9".
25		One-byte variable value from "0" to "9".
26		One-byte variable value from "0" to "9".
27		One-byte variable value from "0" to "9".
28	Separator	Fixed value of $2C_H$ (",")
29	Time zone hours	Fixed dummy value of 30_H ("0")
30		Fixed dummy value of 30_H ("0")
31	Separator	Fixed value of $2C_H$ (",")
32	Time zone minutes	Fixed dummy value of 30_H ("0")
33		Fixed dummy value of 30_H ("0")
34	Checksum separator	Fixed value of $2A_H$ ("*")
35	Checksum	XOR all bytes between "$" and "*", not including these characters.
36		*Note: Must be calculated whenever the frame contents changes.*
37	Message termination	Fixed value of $0D_H$ ([CR])
38		Fixed value of $0A_H$ ([LF])

1PPS Status Information

The status (quality) of the 1PPS signal depends on the reception of the synchronization message and the related multiframe count. **Figure 122** shows how the status is reported.

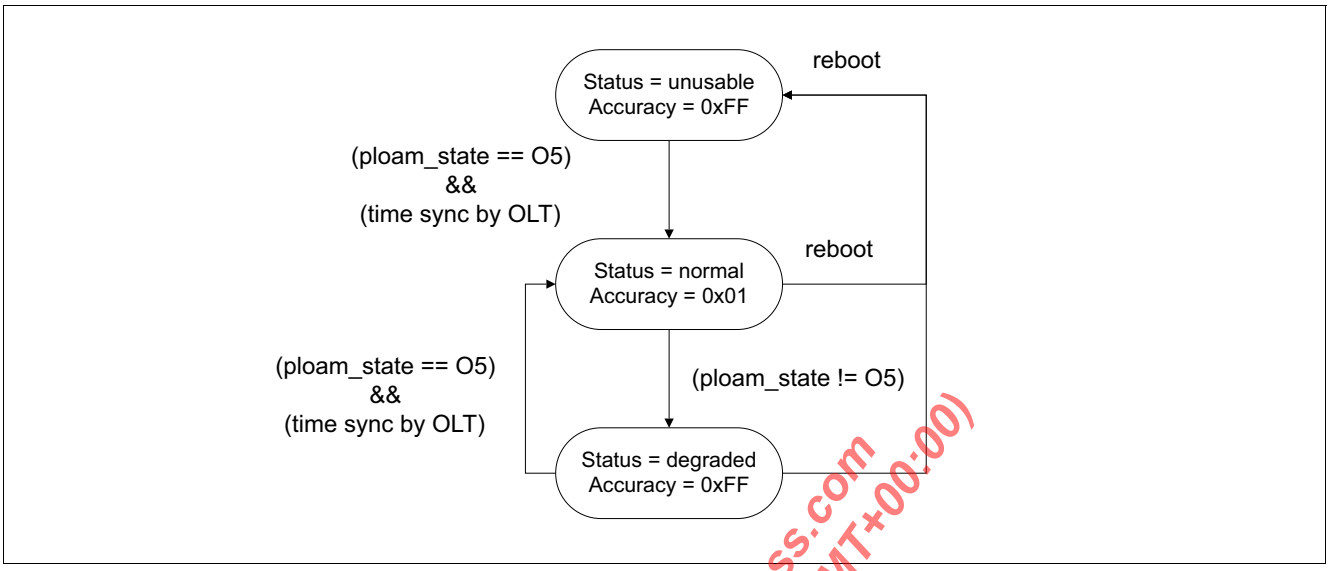


Figure 122 1PPS Status Information

Actions

Get, set

Notifications

This managed entity does not generate notifications.

4.20.6.3 Network Address

ME class: 137

This managed entity is created by the OLT or by the ONU itself and associates a network address with security methods required to access a server. It is conditionally required for ONUs that support VoIP service. The address can represent a URL, a fully qualified path, or an IP address. The address is represented as an ASCII string.

The implementation of this managed entity does not require hardware-related programming.

This managed entity type can be used (for example) by one or more of these managed entities:

- VoIP Configuration Data
- MGC Configuration Data
- SIP Agent Configuration Data
- SIP User Data

These managed entities are beyond the scope of this implementation description.

Managed Entity ID

2 bytes, read, write, set-by-create

This attribute uniquely identifies each instance of this managed entity. Instances of this managed entity created autonomously by the ONU have IDs in the range from 0x0000 to 0x7FFF. Instances created by the OLT have IDs in the range from 0x8000 to 0xFFFE. The value 0xFFFF is reserved.

Security Pointer

2 bytes, read, write, set-by-create

This is a pointer to a **Authentication Security Method** managed entity. This indicates the user name and password to be used when retrieving the network address indicated by this ME. A null pointer indicates that security attributes are not defined for this **Network Address**.

Address Pointer

2 bytes, read, write, set-by-create

This is a pointer to a **Large String** managed entity that presents the network address in ASCII format. It may contain a fully qualified domain name, URI, or IP address. A null pointer indicates that no network address is defined.

Actions

Create, delete, get, set.

4.20.6.4 Authentication Security Method

ME class: 148

This managed entity is created by the OLT.

The authentication security method defines the user ID/password configuration to establish a session between a client and server. This object may be used in the role of the client or server. An instance of this managed entity is created by the OLT if authenticated communication is necessary.

A template implementation is available for reference, the functional implementation is left open for vendor-specific solutions.

Managed Entity ID

2 bytes, read, write, set-by-create

This attribute uniquely identifies each instance of this managed entity.

The value 0xFFFF is reserved.

Validation Scheme

1 byte, read, write

This attribute defines the validation scheme to be applied. Valid code points are:

- 0: No validation
- 1: MD5 digest authentication as defined in IETF RFC 2617.
- 3: Validate using basic authentication as defined in RFC 2617

Other settings must be refused.

Username 1, Username 2

25 bytes, read, write

Each of these two attributes holds an ASCII string that represents the username. Depending on the string length, these attributes are used as follows:

- 1 to 25 characters: Use Username 1 only, fill with null characters. Set Username 2 to all null characters.
- 26 to 50 characters: Use Username 1 to carry the first 25 characters. Use Username 2 to carry the remaining characters, fill with null characters.

Password

25 bytes, read, write

Null-terminated character string, up to 25 characters long.

Realm

25 bytes, read, write

This string attribute specifies the realm used in digest authentication. If the string is shorter than 25 bytes, it must be null terminated.

Actions

Create, delete, get, set.

Notifications

This managed entity does not generate notifications.

4.20.6.5 Large String

ME class: 157

This managed entity holds character strings longer than 25 bytes, up to 375 bytes. It is maintained in up to 15 parts, each part containing 25 bytes. When the final part contains fewer than 25 bytes, it is terminated by at least one null byte. No hardware programming is required to implement this managed entity.

This managed entity is restricted to printable characters.

To cover non-printable characters, the **Octet String** managed entity is provided.

Managed Entity ID

2 bytes, read, write, set-by-create.

This attribute uniquely identifies each instance of this managed entity. The value 0xFFFF is reserved. When the large string is to be used as an IPv6 address, the value 0 is also reserved. The OLT must create large string MEs starting at 1 (or 0), and numbering upwards. The ONU must create large string MEs starting at 65534 (0xFFFE) and numbering downwards.

Number of Parts

1 byte, read, write.

This attribute specifies the number of non-empty parts that form the large string.

This attribute defaults to 0 to indicate no large string content is defined.

Part <n>

25 bytes per part attribute, read, write.

For n = 1 to 15, these attributes are used to define the string.

Actions

Create, delete, get, set.

Notifications

This managed entity generates the following notifications:

AVC #1: Number of parts changed

AVC #2: Part 1 changed

…

AVC #16: Part 15 changed

4.20.6.6 Octet String

ME class: 307

Instances of this managed entity are created and deleted by the OLT. To use this managed entity, the OLT instantiates the Octet String managed entity and then points to the created managed entity from other managed entity instances.

The Octet String is modeled on the **Large String** managed entity. The **Large String** is constrained to printable characters because it uses null as a trailing delimiter. The octet string has a length attribute and is therefore suitable for arbitrary sequences of bytes.

Note: This managed entity is not part of the SDK delivery and must be added by the system vendor, if required.

Managed Entity ID

2 bytes, read, write, set-by-create.

This attribute uniquely identifies each instance of this managed entity. The values 0x0000 and 0xFFFF are reserved.

Length

2 bytes, read, write

This attribute specifies the number of octets that comprise the sequence of octets. This attribute defaults to 0 to indicate no octet string is defined. The maximum value of this attribute is 375 (15 parts, 25 bytes each)

Part <n>

25 bytes, read, write (per part attribute)

For n = 1 to 15, these attributes are used to define the string.

These 15 attributes are identical. The octet string is simply divided into as many parts as necessary, starting at part 1 and left justified.

Actions

Create, delete, get, set.

Notifications

This managed entity does not generate notifications.

4.20.6.7 Threshold Data 1

ME class: 273

The threshold data is partitioned into two managed entities. An instance of this managed entity, together with an optional instance of the **Threshold Data 2** managed entity, contains threshold values for counters in performance monitoring history data (PMHD) managed entities.

The thresholds are used to define when a threshold crossing alert (TCA) must be sent to the OLT. No hardware programming is required to implement this managed entity. Setting the threshold to a value of 0000_H or $FFFF_H$, as well as to a value above the maximum counting value of a counter, disables the threshold.

Managed Entity ID

2 bytes, read, set-by-create.

This attribute uniquely identifies each instance of this managed entity. The value is assigned by the OLT during creation.

Threshold Value 1 … 7

4 bytes, read, write, set-by-create (per attribute)

There are 7 attributes, numbered from 1 to 7.

Actions

Create, delete, get, set.

Notifications

This managed entity does not generate notifications.

4.20.6.8 Threshold Data 2

ME class: 274

This managed entity is used in conjunction with a **Threshold Data 1** managed entity of the number of threshold values to be defined exceeds seven.

No hardware programming is required to implement this managed entity. Setting the threshold to a value of 0000_H or $FFFF_H$, as well as to a value above the maximum counting value of a counter, disables the threshold.

Managed Entity ID

2 bytes, read, set-by-create.

This attribute uniquely identifies each instance of this managed entity. The value is assigned by the OLT during creation to be identical with the related **Threshold Data 1** managed entity.

Threshold Value 8 … 14

4 bytes, read, write, set-by-create (per attribute)

There are 7 attributes, numbered from 8 to 14.

Actions

Create, delete, get, set.

Notifications

This managed entity does not generate notifications.

4.20.6.9 Threshold Data 64-bit

ME class: 426

This managed entity is created by the OLT to provide threshold definitions for 64-bit counters. It provides 14 Threshold Value attributes (numbered 1 to 14). Each attribute provides one threshold value. The size of each attribute is 8 byte.

Managed Entity ID

2 bytes, read, set-by-create.

This attribute uniquely identifies each instance of this managed entity. The value is assigned by the OLT during creation.

Threshold Value 1 … 14

8 bytes, read, write (per attribute)

There are 14 attributes, numbered from 1 to 14.

Actions

Create, delete, get, set.

Notifications

This managed entity does not generate notifications.

4.20.6.10 OMCI

ME class: 287

This managed entity describes the ONU general level of support for OMCI managed entities and messages. It does not require hardware programming and is not part of the MIB upload.

The managed entity data is created on-the-fly each time the OLT performs a read operation, based on the available managed entities.

It is not required to define this in the MIB initialization file, but may be done for debugging purposes. It may also be manually created by using the `omci mec` command-line option.

Manage Entity ID

2 bytes, read-only

This attribute identifies the instance of this managed entity. There is only one instance, the value is 0.

Managed Entity Type Table

N x 2 bytes, read-only

This attribute lists the managed entity classes supported by the ONU. Each entry contains the managed entity class value (refer to **[56]**, table 11.2.4-1) of a managed entity type. N is the number of entries in the list.

Message Type Table

M x 2 bytes, read-only

This attribute is a list of message types supported by the ONU. Each entry contains the message type of an OMCI message (refer to **[56]**, table 11.2.2-1). M is the number of entries in the list.

Actions

Get, get next

Notifications

This managed entity does not generate notifications.

4.20.6.11 Managed Entity

ME class: 288

This managed entity describes the properties of a single other managed entity. The programming is related to the OMCI handling itself and does not require hardware interaction.

This managed entity is not included in the MIB upload.

Manage Entity ID

2 bytes, read-only

This attribute uniquely identifies each instance of this managed entity. Its value is equal to the managed entity type value, and is the same as the code found in the managed entity type table attribute of the OMCI managed entity and table 11.2.4-1 in [56].

Name

25 bytes, read-only

This attribute contains a 25 byte ASCII coded string that represents the managed entity's name. Strings shorter than 25 bytes are padded with null characters (0x00).

Attributes Table

X x 2 bytes, read-only

This table contains pointers to the **Attribute** managed entities that describe each of this managed entity's attributes.

Access

1 byte, read-only

This attribute defines who creates the managed entity.

- 1: Created by the ONU
- 2: Created by the OLT
- 3: Created by the ONU or by the OLT

Alarms Table

Y x 1 byte, read-only

This attribute lists the alarm codes that are supported.

AVC Table

Z x 1 byte, read-only

This attribute lists the AVCs (Attribute Value Change notifications) that are supported

Actions

4 bytes, read-only

This attribute lists the action codes supported on this object, formatted as a bit map. The action codes are the Message Type (MT) values from Table 11.2.2-1 in [56]. The LSB represents action 0, and so on.

Instances Table

V x 2 bytes, read-only

This attribute is a list of pointers to all instances of this managed entity.

Support

1 byte, read-only

This attribute represents the support capability of this managed entity in the ONU's implementation.

- 1: Supported
- 2: Not supported
  OMCI returns an error code, if accessed.
- 3: Partially supported
- 4: Ignored (reference template implementation)
  The OMCI handling is supported, but the underlying functionality is not.

Actions

Get, get next

Notifications

This managed entity does not generate notifications.

4.20.6.12 Attribute

ME class: 289

This managed entity describes the properties of a single attribute that belongs to a certain other managed entity. The programming is related to the OMCI handling itself and does not require hardware interaction.

This managed entity is not included in the MIB upload.

Note: The reference implementation supports the “Name” and “Size” attributes only.

Manage Entity ID

2 bytes, read-only

This attribute uniquely identifies each instance of this managed entity. This number is the same as the one that appears in the attributes table in the managed entity. Only one instance of each unique attribute need be created. The ONU can assign attribute numbering as it pleases. However, it is suggested that the numbering follows a scheme that makes it well readable.

Name

25 bytes, read-only

This attribute contains a 25 byte ASCII coded string that represents the attribute's name. Strings shorter than 25 bytes are padded with null characters (0x00).

Size

2 bytes, read-only

This attribute contains the size of the attribute, in bytes. The value 0 indicates that the attribute can have a variable or unknown size.

Access

1 byte, read-only

This attribute represents the OMCI access characteristics of the attribute. The following code points are defined:

- 1: Read-only
- 2: Write-only
- 3: Read and write
- 4: Read, set-by-create
- 5: Write, set-by-create
- 6: Read, write, set-by-create

Format

1 byte, read-only

This attribute represents the format of the attribute. The following code points are defined:

- 1: Pointer (to a managed entity ID)
- 2: Bit filed
- 3: Signed integer
- 4: Unsigned integer
- 5: String
- 6: Enumeration (of a defined set of code points)
- 7: Table

Lower Limit

4 bytes, read-only

This attribute provides the lowest value for the attribute. This is valid for numerical types only (pointers, signed integers, unsigned integers). For attributes smaller than 4 bytes, the desired numeric value is expressed in a 4 byte representation (for example, the 2s complement 1 byte integer 0xFE is expressed as 0xFFFF FFFE; the unsigned 1 byte integer 0xFE is expressed as 0x0000 00FE).

Upper Limit

4 bytes, read-only

This attribute provides the highest value for the attribute. It has the same validity and format as the **Lower Limit** attribute.

Bit Field

4 bytes, read-only

This attribute is a mask of the supported bits in a bit field attribute, valid for bit field type only. A “1” in any position indicates that its code point is supported, while “0” indicates that it is not supported. For bit fields smaller than 4 bytes, the attribute is aligned at the least significant end of the mask.

Code Points Table

Q x 2 bytes, read-only

This attribute lists the code points that are supported by an enumerated attribute.

Support

1 byte, read-only

This represents the support capability of this attribute in the ONU's implementation.

- 1: Supported
- 2: Not supported
  OMCI returns an error code, if accessed.
- 3: Partially supported
- 4: Ignored (reference template implementation)
  The OMCI handling is supported, but the underlying functionality is not.

Actions

Get, get next

Notifications

This managed entity does not generate notifications.

4.20.6.13 Openflow Configuration Data

ME class: 439

This managed entity contains the configuration data whose underlying transport method is OpenFlow. Instances of this managed entity are created and deleted by the OLT.

An instance of this managed entity is associated with each instance per OpenFlow transportation channel. There might be more than one OpenFlow transportation channel per ONU.

Note: This managed entity is not part of the SDK delivery and must be added by the system vendor, if required.

Managed Entity ID

2 bytes, read-only, set-by-create

This attribute uniquely identifies each instance of this managed entity and is set by the OLT through creation.

TP Type

1 byte, read, write, set-by-create

This attribute specifies the type of ANI-side termination point associated with this managed entity. The only supported type is:

- 1: GEM ITP

TP Pointer

2 bytes, read, write, set-by-create

This attribute points to the instance of the TP associated with this OpenFlow config data. The type of the associated TP is determined by the TP type attribute.

Version

1 byte, read, write, set-by-create

This integer attribute reports the version of the OpenFlow protocol in use.

The ONU should deny an attempt by the OLT to set or create a value that it does not support.

The value 0 indicates that no particular version is specified.

D/S OpenFlow Message

N x 24 bytes, read, write

This attribute specifies the D/S OpenFlow message which is carried over the OMCC channel.

D/S Forwarding Control

1 byte, read, write

This boolean attribute indicates the current D/S OpenFlow message is ready to be sent (true) or not.

The default value is false.

D/S Receiving Status

1 byte, read-only

This boolean attribute indicates the ONU is ready to accept a new downstream packet (true) or not.

The default value is false.

U/S OpenFlow Message

N x 24 bytes, read, write

This attribute specifies the U/S OpenFlow message which is carried over the OMCC channel.

U/S OpenFlow Control

1 byte, read, write

This boolean attribute controls the current U/S OpenFlow message is safely received (true) or not.

The default value is false.

U/S Forwarding Status

1 byte, read-only

This boolean attribute reports the current U/S OpenFlow message is ready to be sent (true) or not.

The default value is false.

Circuit ID

N x 24 bytes, read, write

This attribute identifies the first access information of the user.

Remote ID

N x 24 bytes, read, write

This attribute identifies the second access information of the user as an addition identifier to circuit ID.

Administrative State

1 byte, read, write

This attribute locks (1) and unlocks (0) the functions performed by this managed entity.

Operational State

1 byte, read-only

This attribute reports whether the managed entity is currently capable of performing its function. Valid values are enabled (0) and disabled (1).

Actions

Create, delete, get, get next, set.

Notifications

AVC # 6: Downstream receiving status.

AVC #9: Upstream forwarding status.

AVC # 13: Operational state.

4.20.6.14 General Purpose Buffer

ME class: 308

This managed entity is created by the OLT when needed to store the results of an operation, such as a test command, that needs to return a block of data of indeterminate size. The buffer is retrieved with get next operations, since its size is not known a priori. An instance of this managed entity is created and deleted by the OLT, and typically made known to an ONU managed entity or to an action through a pointer.

The managed entity is defined as generically as possible, such that it can be used for other applications that may not initially be apparent, such as logging. The format of its content is specific to each application, and is documented there.

Through a pointer, the OLT may associate a general purpose buffer with a managed entity and/or an operation that has a need to create large or indeterminate blocks of data for subsequent upload.

The general purpose buffer is neither captured in a MIB upload, nor retained in a non-volatile ONU memory.

Note: This managed entity is not part of the SDK delivery and must be added by the system vendor, if required.

Managed Entity ID

2 bytes, read-only, set-by-create

A unique ID is assigned by the OLT through creation.

Maximum Size

4 bytes, read, write, set-by-create

The ONU determines the actual size of the buffer table in the process of capturing the data directed to it. The maximum size attribute permits the OLT to restrict the maximum size of the buffer table. The value 0 indicates that the OLT imposes no limit on the size. It is recognized that ONU implementations will impose their own limits. The ONU will not create a buffer table larger than the value of this attribute. If the ONU cannot allocate enough memory to accommodate this size, it should deny the managed entity create action or a write operation that attempts to expand an existing managed entity.

Buffer Table

n bytes, read-only

This attribute is an octet string that contains the result of some operation performed on the ONU. The exact content depends on the operation, and is documented with the definition of each operation.

Actions

Create, delete, get, get next

Notifications

AVC #2: Buffer table

4.20.6.15 File Transfer Controller

ME class: 318

This optional managed entity allows file transfers to be conducted out of band. It is intended to facilitate software image download, but may be used for other file transfer applications as well.

One instance of this managed entity exists in an ONU that supports out-of-band file transfer.

Note: This managed entity is not part of the SDK delivery and must be added by the system vendor, if required.

Managed Entity ID

2 bytes, read-only

This attribute uniquely identifies the instance of this managed entity. There is only one instance, created by the ONU, the value is 0.

Supported Transfer Protocols

2 bytes, read-only

File Type

2 bytes, read, write

File Instance

2 bytes, read, write

Local File Name Pointer

2 bytes, read, write

Network Address Pointer

2 bytes, read, write

File Transfer Trigger

1 byte, read, write

File Transfer Status

1 byte, read-only

GEM IWTP Pointer

2 bytes, read, write

VLAN

2 bytes, read, write

File Size

4 bytes, read, write

Directory Listing Table

N x 1 byte, read-only

Actions

Get, set

Notifications

AVC #7: File transfer status

AVC #11: Directory listing table

4.20.6.16 Generic Status Portal

ME class: 330

The generic status portal managed entity provides a way for the OLT to discover the status and configuration information of a non-OMCI management domain within an ONU. The non-OMCI management domain is indicated by the virtual Ethernet interface point associated with this generic status portal. The generic status portal ME uses two table attributes to convey status and configuration from a non-OMCI managed domain to the OMCI. Each of these attributes uses an XML document to present this information. These XML documents are not required to be understood by the OLT or EMS. It is possible to use the XML document schema to create tools which parse and interpret the contents of the document..

One instance of this ME is created by the OLT for each separate non-OMCI management domain whose information is desired to be visible.

Note: This managed entity is provided as a reference only, the functional implementation must be added by the system vendor.

Managed Entity ID

2 bytes, read-only

This attribute uniquely identifies each instance of this managed entity. Through an identical ID, the generic status portal managed entity is implicitly linked to an instance of the virtual Ethernet interface point managed entity.

Status Document Table

N x 1 byte, read-only

This attribute is used to pass a textual representation of the non-OMCI managed domain status back to the OLT. The contents are vendor-specific and formatted as an XML document. The first element of the document must contain an XML declaration indicating the version of XML and encoding used in the remainder of the document. The second element of the document must contain a schema reference to the vendor-supplied schema used by the remainder of the document. The get, get next sequence must be used with this attribute since its size is unspecified.

Configuration Document Table

M x 1 byte, read, write

This attribute is used to pass a textual representation of the non-OMCI managed domain configuration back to the OLT. The contents are vendor-specific and formatted as an XML document. The first element of the document must contain an XML declaration indicating the version of XML and encoding used in the remainder of the document. The second element of the document must contain a schema reference to the vendor-supplied schema used by the remainder of the document. The get, get next sequence must be used with this attribute since its size is unspecified.

AVC Report Rate

1 byte, read, write, set-by-create

This attribute governs the rate at which the generic status portal generates attribute value change notifications. The default value 0 disables AVCs, while the highest value 3, which may be useful for debugging, generates an

AVC on every change seen in the non-OMCI management domain. As a guideline, the value 1 should collect changes into not more than one notification in ten minutes, while value 2 should generate an AVC not more than once per second.

Actions

Create, delete, get, get next

Notifications

AVC #1: Status document table

AVC #2: Configuration document table

4.20.7 Miscellaneous Managed Entities

These managed entities are defined by ITU-T G.988 **[56]** in chapter 9.13.

4.20.7.1 PPTP LCT UNI

ME class: 83

This managed entity models debug access to the ONU from any physical UNI port. The implementation uses exception handling based on destination MAC address detection. See **Figure 123** for details.

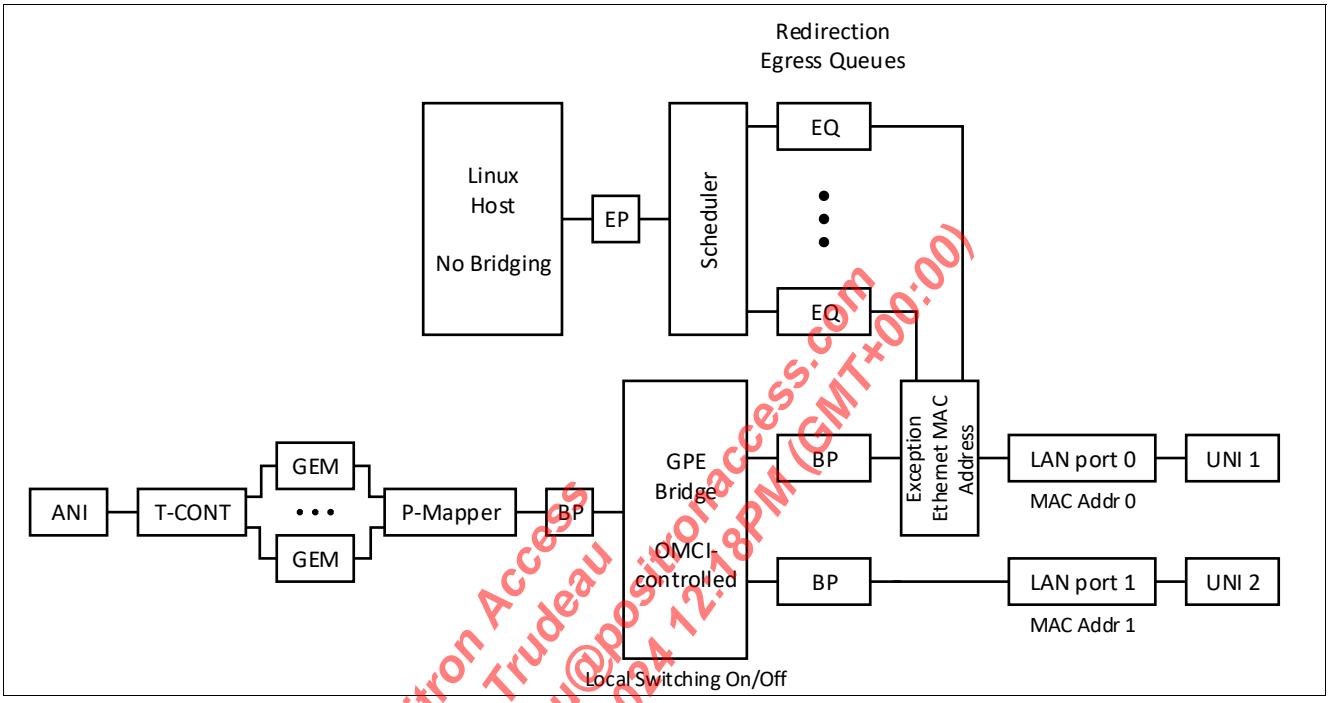


Figure 123 PPTP LCT UNI Internal Structure

One of the UNI interfaces can be used as the LCT port with an individual MAC address assigned. In this scenario, a dedicated network device is created. This network device receives automatically the unique MAC address. For the configuration of these interfaces, the standard Linux network tools (`iproute2`) or libraries may be used. The exact network device relationship is presented in **OMCI Bridge Mapping to Linux**. The LCT network device is named `eth0_[n]_1_lct` where `n` is the port number. For example, `eth0_0_1_lct` is the LCT interface for the first LAN port and `eth0_1_1_lct` would be the LCT interface for the second UNI port. The LCT network interface creation can be enabled or disabled by using the Linux kernel device tree.

To change the interface from one UNI port to another, the UCI network configuration can be changed.

For example, change the `ifname` and `lct_port` values from:

```
config interface 'lct'
        option ifname 'eth0_0_1_lct'
        option lct_port '1'
```

to:

```
config interface 'lct'
        option ifname 'eth0_1_1_lct'
        option lct_port '2'
```

The PPTP LCT interface is configured in the MIB initialization file and not included in the MIB upload. The managed entity ID value given in the MIB initialization file is modified by the selection of `lct_port`, if a value other than 1 is selected.

Managed Entity ID

2 bytes, read-only, set-by-create.

This attribute uniquely identifies each instance of this managed entity. The value of 0 is used to control all LCT interfaces commonly.

Note: Only a single LCT instance is currently supported.

Administrative State

1 byte, read, write.

This attribute locks (disables) or unlocks (enables) the function of the related interface.

- 0: The interface is unlocked and is able to transmit data.
- 1: The interface is locked, no data transmission is possible.

When the administrative state is set to lock, debug access through all physical or logical means is blocked, except that the operation of a possible ONU remote debug ME is not affected. Administrative lock of ME instance 0 overrides administrative lock of any other PPTP LCT UNIs that may exist.

To enable and disable the LCT interface, the Linux `ip` function is used:

```
ip link set dev eth0_0_<0|1>_lct <up|down>
```

To control the interface from the C code, there is a netlink command available in the `pon_libnl` library:

```
int rtnl_link_change(struct nl_sock *sk, struct rtnl_link *orig, struct rtnl_link
*changes, int flags)
```

The default setting of the administrative state is defined in the UCI configuration file `/etc/config/omci` in lct_disable.

The value shall be set to locked (1) by default if access from UNI side through the LCT network interface shall be granted only under OLT control. If the OLT does not control the LCT interface, the value must be set to unlocked (0), to use the LCT access from UNI side. The latter is the default setting.

Actions

Get, set

Notifications

This managed entity does not generate notifications.

4.20.7.2 Time Status Message

ME class: 440

This managed entity provides status and characterization information for the time transmitting node and its grandmaster. When time synchronization is included in the ONU functionality, this managed entity is automatically created from the MIB initialization file. This includes the default values to be set.

This managed entity provides an interface between the OLT and the SyncE and PTP functions locally provided on the OLT. This requires the `synce` and `ptp4l` daemons to be started before the `omcid` is initialized.

Note: This managed entity is not available and planned for future implementation in the reference OMCI stack.

Managed Entity ID

2 bytes, read-only.

This attribute uniquely identifies each instance of this managed entity and is fixed to 0.

There is only a single instance of this managed entity present within an ONU.

Domain Number

1 byte, read, write.

This attribute specifies the `domainNumber`, using the format of IEEE 1588 clause 7.1: *The domain Number of a domain is an integer in the closed range 0 to 255.*

The value is read from the `ptp4l` daemon. The changes are done by the OLT and are communicated from `omcid` to `ptp4l` over UDS, using the management TLV message format.

The default value is 0.

Flag Field

1 byte, read, write.

Table 68 shows that this attribute specifies the flag field as an 8-bit mask. A bit value of 1 means true.

Table 68 PTP Flag Field

Bit	Function	Software Handling
0 (LSB)	Leap61; add one leap second	Received by `ptp4l` from the grandmaster clock server, reported to the OLT.
1	Leap59; remove one leap second	
2	`currentUtcOffsetValid`	
3	PTP timescale	
4	Time traceable	
5	Frequency traceable	
6	Reserved	Ignore on reception, send as 0.
7 (MSB)	Reserved	Ignore on reception, send as 0.

The bit values are read from the `ptp4l` daemon. The changes are done by the OLT and are communicated from `omcid` to `ptp4l` over UDS, using the management TLV message format.

The default value is 0x00.

Current UTC Offset

2 bytes, read, write.

This attribute specifies the UTC offset value between the TAI and UTC timescales (UTC Offset = TAI - UTC), as specified in IEEE 1588 clause 7.2.3: *The PTP epoch is 1 January 1970 00:00:00 TAI, which is 31 December 1969 23:59:51.999918 UTC.*

The value is read from the `ptp4l` daemon. The changes are done by the OLT and are communicated from `omcid` to `ptp4l` over UDS, using the management TLV message format.

The default value is defined in the MIB initialization file.

Priority 1

1 byte, read, write.

This attribute specifies the value as defined in IEEE 1588 clause 7.6.2.2: *The attribute priority1 is used in the execution of the best master clock algorithm. Lower values take precedence. The initialization value of priority1 is specified in the applicable PTP Profile. The value of priority1 must be configurable to any value in the range 0 to 255, unless restricted by limits established by the applicable PTP Profile.*

The value is read from the `ptp4l` daemon. The changes are done by the OLT and are communicated from `omcid` to `ptp4l` over UDS, using the management TLV message format.

The default value is defined in the MIB initialization file.

Clock Class

1 byte, read, write.

This attribute specifies the clockClass information denoting the traceability of the time distributed by the grandmaster clock. as specified in IEEE 1588 clause 7.6.2.4: *The clockClass attribute of an Ordinary Clock or Boundary Clock denotes the traceability, synchronization state and expected performance of the time or frequency distributed by the Grandmaster PTP Instance. The clockClass is one of the attributes that characterizes a PTP Instance for the purpose of the best master clock algorithm.*

The value is read from the `ptp4l` daemon. The changes are done by the OLT and are communicated from `omcid` to `ptp4l` over UDS, using the management TLV message format.

The default value is defined in the MIB initialization file.

Accuracy

1 byte, read, write.

This attribute value indicates the expected accuracy of a clock when it is the grandmaster, as it is specified in IEEE 1588 clause 7.6.2.5: *The clockAccuracy is one of the attributes that characterizes a PTP Instance for the purpose of the best master clock algorithm.*

The value is read from the `ptp4l` daemon. The changes are done by the OLT and are communicated from `omcid` to `ptp4l` over UDS, using the management TLV message format.

The default value is defined in the MIB initialization file.

offsetScaledLogVariance

2 bytes, read, write.

This attribute specifies the estimate of the time variance, as specified in IEEE 1588 clause 7.6.3: *Each such PTP Instance must maintain an estimate, the offsetScaledLogVariance, of its inherent precision. This is the precision of the timestamps included in PTP messages issued by the PTP Instance when it is not synchronized to the Local PTP Clock of another PTP Instance using the protocol. The offsetScaledLogVariance depends on both the stability of the Local Clock of the PTP instance and any error 2 introduced in the time-stamping process.*

The value is read from the `ptp4l` daemon. The changes are done by the OLT and are communicated from `omcid` to `ptp4l` over UDS, using the management TLV message format.

The default value is defined in the MIB initialization file.

Priority 2

1 byte, read, write.

This attribute specifies the value as defined in IEEE 1588 clause 7.6.2.3: *The attribute priority2 is used in the execution of the best master clock algorithm. Lower values take precedence. The initialization value of priority2 is specified in the applicable PTP Profile. The value of priority2 must be configurable to any value in the range 0 to 255, unless restricted by limits established by the applicable PTP Profile.*

The value is read from the `ptp4l` daemon. The changes are done by the OLT and are communicated from `omcid` to `ptp4l` over UDS, using the management TLV message format.

The default value is defined in the MIB initialization file.

Grandmaster ID

8 bytes, read, write.

This attribute specifies the clockIdentity attribute of the grandmaster, taken from the IEEE EUI-64 individual assigned numbers.

The value is read from the `ptp4l` daemon. The changes are done by the OLT and are communicated from `omcid` to `ptp4l` over UDS, using the management TLV message format.

The default value is defined in the MIB initialization file.

Steps Removed

2 bytes, read, write.

This attribute specifies the number of boundary clocks between the local clock and the master.

The value is read from the `ptp4l` daemon. The changes are done by the OLT and are communicated from `omcid` to `ptp4l` over UDS, using the management TLV message format.

The default value is defined in the MIB initialization file.

Time source

1 byte, read, write.

This attribute specifies the source of time used by the grandmaster clock, as it is specified in IEEE 1588 clause 7.6.2.6 (clock accuracy).

The value is read from the `ptp4l` daemon. The changes are done by the OLT and are communicated from `omcid` to `ptp4l` over UDS, using the management TLV message format.

The default value is defined in the MIB initialization file.

Actions

Get, set

Notifications

The following notifications must be sent by this managed entity:

- Alarm #0: Clock Unlock
  When the clock PLL is unable to work properly, it must be able to automatically report the clock unlock alarm. The alarm status information is retrieved from the synced (netlink message).
- Alarm #1: ESMC Loss
  When no valid ESMC packet is received during 5 seconds, the ESMC loss is detected and the ESMC loss alarm is sent. The alarm status information is retrieved from the synced (netlink message).
- Alarm #2: Time Unlock
  When the time PLL cannot be locked, it must be able to automatically report the time unlock alarm.

The time unlock alarm is set each time the PON ToD `PON IP`, `tod_status_get` or XGMAC ToD `phc2sys` is invalid. The alarm is cleared when both are valid.

4.20.8 Security Configuration

There are security enhancements available in the newer ITU PON versions covered within this section.

4.20.8.1 Enhanced Security Control

ME class: 332

This managed entity contains the capabilities, parameters and controls of enhanced PON security features when they are negotiated via the OMCI. The attributes in this managed entity are intended to be used to implement a symmetric-key-based three-step authentication process, as described in the supplemental information section of ITU-T G.988.
This managed entity is created by the ONU autonomously and related to the ONU as a whole (**ONU-G**/**ONU2-G**). This managed entity is created by the ONU when the ONU-G and ONU2-G managed entities have been created. It is available in XG-PON, NG-PON2, or XGS-PON operation mode only and is not used on G-PON mode.

These attribute values provided by this managed entity must be implemented as detailed hereafter:

- **OLT Cryptographic Capabilities**
- **OLT Random Challenge Table**
- **OLT Challenge Status**
- **ONU Selected Cryptographic Capability**
- **ONU Random Challenge Table**
- **ONU Authentication Result Table**
- **OLT Authentication Result Table**
- **OLT Result Status**
- **ONU Authentication Status**
- **Master Session Key Name**
- **Broadcast Key Table**

When a system uses the 802.1X authentication instead of the OMCI method provided here, the only applicable attribute of this managed entity is the **Broadcast Key Table**.
Except for the **Broadcast Key Table**, the use of the attributes are optional and used only when the OLT requires it.

Figure 124 shows the authentication flow.

Note: The current reference implementation covers only the Broadcast Key Table handling.

Managed Entity ID

2 bytes, read-only

The managed entity ID is set to a fixed value of 0 (the same as for the other central managed entities).

OLT Cryptographic Capabilities

16 bytes, write-only

This is an information by the OLT regarding which cryptographic capabilities are provided. Each capability is advertised by setting the related bit in a bit mask.

The options are:

- Bit 0 (LSB): AES-CMAC-128
  This is mandatory and must be supported by all OLTs.
  This is also the only mode supported by the hardware.
- Bit 1: HMAC-SHA-256
- Bit 2: HMAC-SHA-512
- Bit 3 to bit 127: Reserved, to be set to 0.

Each time the OLT writes to this attribute, the value of the **OLT Challenge Status** attribute is set to false.

The value received in this attribute is evaluated for the response given in the attribute **ONU Selected Cryptographic Capability**.

OLT Random Challenge Table

N x 17 bytes, read, write

This attribute specifies the random challenge value issued by the OLT during authentication step 1.

It is structured as a table, with each entry being 17 bytes. The first byte is the table row number, starting at 1, and the remaining 16 bytes are the contents of the entry.

The OLT challenge is the concatenation of all 16-byte content fields. The OLT writes all the entries in the table, and then triggers the ONU processing of the entire table by setting the **OLT Challenge Status** attribute to true. The table size is known by the maximum index set by the OLT:

```
olt_random_challenge_size = max_index * 16 /* number of bytes */
```

The OLT clears the table with a set operation to row 0.

Each time the OLT writes to this attribute, the value of the **OLT Challenge Status** attribute is set to false.

OLT Challenge Status

1 byte, read, write

When the OLT sets this value from false to true, the contents of the **OLT Random Challenge Table** is sent to the Device1 by using the function `fapi_pon_auth_olt_challenge_set`.

In addition, the contents of **ONU Random Challenge Table** is generated from a random number and sent to the PON IP.

The value of this attribute is set to false each time either the **OLT Cryptographic Capabilities** or the **OLT Random Challenge Table** attribute is written by the OLT.

ONU Selected Cryptographic Capability

1 byte, read-only

The software first checks the cryptographic capabilities of the Device1 by using the `fapi_pon_cap_get` function. The bit values `cry1`, `cry2`, and `cry3` are related to the **OLT Cryptographic Capabilities** coded in bit 0, 1, and 2.

The bit vector returned as `features` must be analyzed at the bit positions defined by `PON_FEATURE_CRY<n>` for matching capabilities.

The software checks for the highest bit position both supported by the OLT and Device1. The numeric value of the bit position is returned to the OLT.

When no match is found, a value of 0 is returned. This is an unexpected error case.

Note: For the initial implementations, it is expected that the OLT only offers the mandatory mode 1 (bit 0) and the ONU only offers `cry1`*. Thus, the returned value is fixed to 1.*

ONU Random Challenge Table

P x 17 bytes, read-only

This attribute specifies the random challenge `ONU_challenge` issued by the ONU during authentication step 2. It is structured as a table, with each entry being 16 bytes of content. The ONU challenge is the concatenation of all 16-byte content fields in the table.

When the OLT triggers a response to be generated using the **OLT Challenge Status** attribute (setting this from false to true), a new random challenge value is requested from the firmware by calling the `fapi_pon_auth_onu_challenge_get` function.

After the random value has been made available, an AVC is generated which signals to the OLT that the challenge is ready, so that the OLT may start a get/get-next sequence to obtain the table contents.

The random value size must be the same as the **OLT Random Challenge Table** size received from the OLT.

ONU Authentication Result Table

Q x 16 bytes, read-only

After the **ONU Random Challenge Table** is sent through `fapi_pon_auth_onu_challenge_set` to the PON IP firmware, the authentication result is calculated. When the calculation is completed, the result is read back from the firmware by using the `fapi_pon_auth_onu_result_get` function. Data availability is indicated by a firmware alarm message (ARR, authentication result ready).

After the data is received from the firmware and stored in the OMCI software, an AVC is generated to inform the OLT that the data is available.

The authentication result is calculated by the firmware according to this rule:

```
ONU_result = SelectedHashFunction (PSK, (ONU_selected_crypto capabilities |
OLT_challenge | ONU_challenge | 0x0000 0000 0000 0000))
```

As the pre-shared key (PSK) in this calculation, the ONU serial number is used (subject to vendor agreement or standardization, may be changed to registration ID). As the serial number size is only 8 byte (64 bit), the LSB part must be padded with zeros to the required key size.

OLT Authentication Result Table

R x 17 bytes, write-only

The OLT write this attribute as a table. When all table rows are sent, the OLT sets the **OLT Result Status** attribute from false to true. Upon this event, the contents of this attribute are sent to the PON IP by using `fapi_pon_auth_olt_result_set`. The table size is known by the maximum index set by the OLT.

OLT Result Status

1 byte, read, write

This value is set to false by the ONU as soon as the first OLT write access to the **OLT Authentication Result Table** is detected.

When the **OLT Authentication Result Table** is received and this attribute value is set to true by the OLT, the contents of this table are sent to the firmware through the `fapi_pon_auth_olt_result_set` function.

ONU Authentication Status

1 byte, read-only

After the **OLT Authentication Result Table** is sent to the PON IP firmware, the `fapi_pon_gpon_status_get` function is used to read back the status (`auth_status`). Data availability is indicated by a firmware alarm message (ASC, authentication status change).

On each status change, an AVC message is sent to the OLT.

Master Session Key Name

16 bytes, read-only

The master session key name is the hash performed on the MSK. This is done by the PON IP firmware and read through the `fapi_pon_auth_onu_msk_hash_get` function.

The hashing algorithm is the same as selected by the **ONU Selected Cryptographic Capability** attribute.

Broadcast Key Table

N x 18 bytes, read, write

This table receives the broadcast encryption key from the OLT. It is stored in rows of 16 byte. The size of the table is defined by the OLT, the key is encrypted with AES-ECB using the KEK. When the table is completely received, it is sent to the Device1 firmware by using the `fapi_pon_auth_onu_bc_key_set` function.

The OLT does not send an explicit indication after the key update is completed.

To detect the completion of the key transmission, the software must check that all key fragments are received, after a first fragment is sent by the OLT. The key fragments are received in any order. The number of required key fragments depends on the selected key size.

The key size is defined by the cryptographic mode selected by the **ONU Selected Cryptographic Capability** attribute.

Effective Key Length

2 bytes, read-only

This attribute specifies the maximum effective length, in bits, of keys generated by the ONU. It is reported based on the selected cryptographic algorithm, as reported by the **ONU Selected Cryptographic Capability**.

The ONU always generates an effective key length as large as possible for the selected encryption mode.

Actions

Get, get current data, set

Notifications

These notifications must be sent by this managed entity:

- Alarm #5: ONU random challenge table (triggered by a change of attribute #5).
  This AVC is sent each time the software creates a new ONU random challenge value.
  This is triggered by a change of the **OLT Challenge Status** from false to true.
- Alarm #6: ONU authentication result table (triggered by a change of attribute #6).
  This AVC is sent each time the **ONU Authentication Result Table** changes its contents.
- Alarm #9: ONU authentication status (triggered by a change of attribute #9).
  This AVC is sent each time the **ONU Authentication Status** value changes.

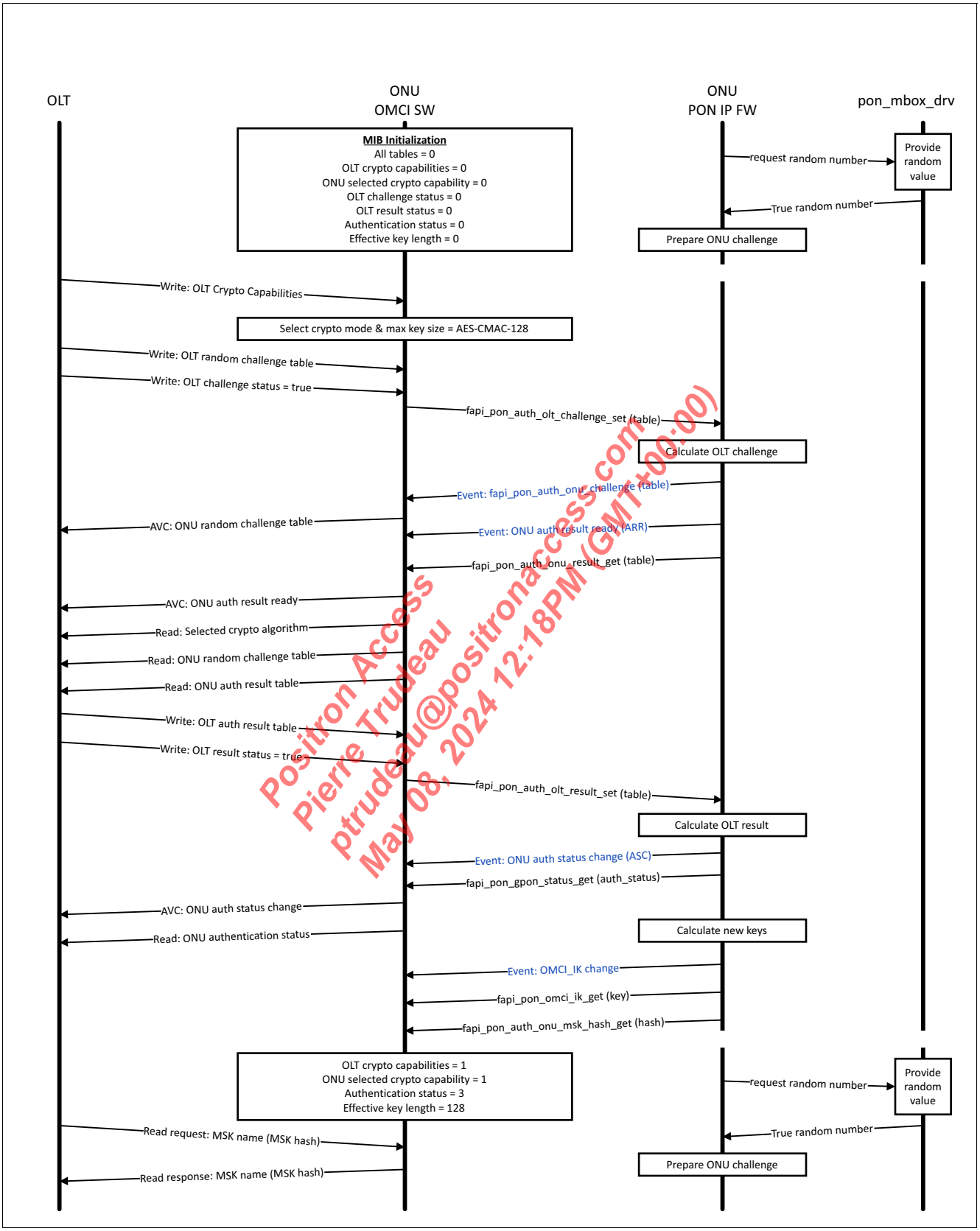


Figure 124 Mutual Authentication Message Flow

4.20.9 XG-PON Enhancements

This chapter describes managed entities which are defined only for XG-PON or XGS-PON implementations. These managed entities are not created if the system is run in ITU-T G.984 or ITU-T G.989 operation mode.

Note: These managed entities is not part of the reference implementation.

4.20.9.1 XG-PON TC Performance Monitoring History Data

ME class: 344

This managed entity is created at the request of the OLT. It is available in XG-PON or XGX-PON operation mode only and is not used on G-PON mode nor in NG-PON2 mode.

The error counters provided by this managed entity should be implemented by using the function as defined by `fapi_pon_xgtc_counters_get`.

Note: This managed entity is not part of the reference implementation.

Managed Entity ID

Through an identical ID, this managed entity is implicitly linked to an instance of the related **ANI-G** managed entity.

Interval End Time

1 byte, read-only

This attribute identifies the most recently finished 15-minute interval and is implemented as in all other PMHD managed entities.

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The managed entities that are pointed to by this value provide alarm thresholds which are used for the monitored counters.

PSBd HEC Error Count

4 bytes, read-only

The number of PSB downstream uncorrected HEC errors in any of the fields of the downstream physical synchronization block is reported by `fapi_pon_xgtc_counters_get` (`psbd_hec_err_uncorr`).

XGTC HEC Error Count

4 bytes, read-only

The number of downstream uncorrected XGTC HEC errors is reported by `fapi_pon_xgtc_counters_get` (`fs_hec_err_uncorr`).

Unknown Profile Count

4 bytes, read-only

This attribute counts the number of grants received whose specified profile was not known to the ONU.

The number of unknown profiles received is reported by `fapi_pon_xgtc_counters_get` (`unknown_profiles`).

Transmitted XGEM Frame Count

4 bytes, read-only

This attribute counts the number of non-idle XGEM frames transmitted. If an SDU is fragmented, each fragment is an XGEM frame and is counted as such.

The number of transmitted XGEM frames is reported by `fapi_pon_xgtc_counters_get` (`tx_frames`)

Transmitted XGEM Frame Fragment Count

4 bytes, read-only

This attribute counts the number of XGEM frames that represent fragmented SDUs, as indicated by the LF bit = 0.

The number of transmitted XGEM frame fragments is reported by `fapi_pon_xgtc_counters_get` (`tx_fragments`)

XGEM HEC Lost Words Count

4 bytes, read-only

This attribute counts the number of four-byte words lost because of an XGEM frame HEC error. In general, all XGTC payload following the error is lost, until the next PSBd event.

The number of lost words is reported by `fapi_pon_xgtc_counters_get` (`lost_words`)

XGEM Key Error Count

4 bytes, read-only

This attribute counts the number of downstream XGEM frames received with an invalid key specification. The key may be invalid for several reasons, this attribute provides the overall count.

The number of errors is reported by `fapi_pon_xgtc_counters_get` (`xgem_key_err`).

XGEM HEC Error Count

4 bytes, read-only

This attribute counts the number of instances of an XGEM frame HEC error.

The number of errors is reported by `fapi_pon_xgtc_counters_get` (`xgem_hec_err_uncorr`)

Transmitted Bytes in Non-idle XGEM Frames Count

8 bytes, read-only

This attribute counts the number of bytes transmitted in non-idle XGEM upstream frames.

The number of errors is reported by `fapi_pon_gem_port_counters_get` (`tx_bytes`)
The counter values are accumulated over all active XGEM ports.

Received Bytes in Non-idle XGEM Frames Count

8 bytes, read-only

This attribute counts the number of bytes received in non-idle XGEM downstream frames.

The number of errors is reported by `fapi_pon_gem_port_counters_get` (`rx_bytes`)
The counter values are accumulated over all active XGEM ports.

LODS Event Count

4 bytes, read-only

This attribute counts the number of LODS events (PLOAM activation state transitions from O5.1 to O6).

Note: TODO: This counter is currently not available and needs to be added to the SW The FW provides TOTAL_LODS_LOF in the GTC_ENHANCED_COUNTERS message.

LODS Event Restored Count

4 bytes, read-only

This attribute counts the number of restored (cleared) LODS events (PLOAM activation state transitions from O6 to O5.1).

LODS Event ONU Reactivation Count

4 bytes, read-only

This attribute counts the number of LODS events resulting in ONU reactivation.

Actions

Create, delete, get, set, get current data

Attribute Value Change Notifications

There are no Attribute Value Changes defined for this managed entity.

Alarm Notifications

The following notifications are sent by this managed entity:

- Alarm #1: PSBd HEC error count (triggered by threshold attribute #1)
- Alarm #2: XGTC HEC error count (triggered by threshold attribute #2)
- Alarm #3: Unknown profile count (triggered by threshold attribute #3)
- Alarm #4: XGEM HEC loss count (triggered by threshold attribute #4)
- Alarm #5: XGEM key errors (triggered by threshold attribute #5)
- Alarm #6: XGEM HEC error count (triggered by threshold attribute #6)

4.20.9.2 XG-PON Downstream Management Performance Monitoring History Data

ME class: 345

This managed entity is created at the request of the OLT. It is available in XG-PON or XGX-PON operation mode only and is not used on G-PON mode.

The error counters provided by this managed entity should be implemented by using the PON library counter functions. The counter size is 32 bit (4 byte). All counters are related to the PLOAM and OMCI reception process in downstream direction.

Note: This managed entity is not part of the reference implementation.

Managed Entity ID

Through an identical ID, this managed entity is implicitly linked to an instance of the related **ANI-G** managed entity.

Interval End Time

1 byte, read-only

This attribute identifies the most recently finished 15-minute interval and is implemented as in all other PMHD managed entities.

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The managed entities that are pointed to by this value provide alarm thresholds which are used for the monitored counters.

PLOAM MIC Error Count

4 bytes, read-only

This attribute counts MIC errors detected in downstream PLOAM messages, either directed to this ONU or broadcast to all ONUs.

The number of PLOAM MIC errors is available through the PON adapter (`pa_management_ds_cnt`, `ploam_mic_errors`)

Downstream PLOAM Message Count

4 bytes, read-only

This attribute counts the number of PLOAM messages received, either directed to this ONU or broadcast to all ONUs. The number of messages to be reported here is the total sum of all counter values as provided by the PON adapter (`pa_management_ds_cnt`, `all_ds`).

Profile Message Count

4 bytes, read-only

This attribute counts the number of profile messages received, either directed to this ONU or broadcast to all ONUs. The value is provided by the PON adapter (`pa_management_ds_cnt`, `profile`).

Ranging Time Message Count

4 bytes, read-only

This attribute counts the number of ranging_time messages received, either directed to this ONU or broadcast to all ONUs. The value is provided by the PON adapter (`pa_management_ds_cnt, ranging_time`).

Deactivate ONU ID Message Count

4 bytes, read-only

This attribute counts the number of deactivate_ONU-ID messages received, either directed to this ONU or broadcast to all ONUs. Deactivate_ONU-ID messages do not reset this counter. The value is provided by the PON adapter (`pa_management_ds_cnt, deactivate_onu`).

Disable Serial Number Message Count

4 bytes, read-only

This attribute counts the number of disable_serial_number messages received, whose serial number specified this ONU. The value is provided by the PON adapter (`pa_management_ds_cnt, disable_serial_number`).

Request Registration Message Count

4 bytes, read-only

This attribute counts the number of request_registration messages received. The value is provided by the PON adapter (`pa_management_ds_cnt, request_registration`).

Assign Alloc ID Message Count

4 bytes, read-only

This attribute counts the number of assign_alloc-ID messages received. The value is provided by the PON adapter (`pa_management_ds_cnt, assign_allocid`).

Key Control Message Count

4 bytes, read-only

This attribute counts the number of key_control messages received, either directed to this ONU or broadcast to all ONUs. The value is provided by the PON adapter (`pa_management_ds_cnt, key_control`).

Sleep Allow Message Count

4 bytes, read-only

This attribute counts the number of sleep_allow messages received, either directed to this ONU or broadcast to all ONUs. The value is provided by the PON adapter (`pa_management_ds_cnt, sleep_allow`).

Baseline OMCI Message Count

4 bytes, read-only

This attribute counts the number of OMCI messages received in the baseline message format. The value is provided by `fapi_pon_omci_counters_get` (`omci_baseline_rx_msg`).

Extended OMCI Message Count

4 bytes, read-only

This attribute counts the number of OMCI messages received in the extended message format. The value is provided by `fapi_pon_omci_counters_get` (`omci_extended_rx_msg`).

Assign ONU ID Message Count

4 bytes, read-only

This attribute counts the number of assign_ONU-ID messages received since the last re-boot. The value is provided by the PON adapter (`pa_management_ds_cnt, assign_onu_id`).

OMCI MIC Error Count

4 bytes, read-only

This attribute counts MIC errors detected in OMCI messages directed to this ONU. The value is provided by `fapi_pon_omci_counters_get` (`omci_mic_err`).

Actions

Create, delete, get, set, get current data

Attribute Value Change Notifications

There are no Attribute Value Changes defined for this managed entity.

Alarm Notifications

The following notifications are sent by this managed entity:

- Alarm #1: PLOAM MIC error count (triggered by threshold attribute #1)
- Alarm #2: OMCI MIC error count (triggered by threshold attribute #2)

4.20.9.3 XG-PON Upstream Management Performance Monitoring History Data

ME class: 346

This managed entity is created at the request of the OLT. It is available in XG-PON or XGX-PON operation mode only and is not used on G-PON mode.

The error counters provided by this managed entity should be implemented by using the PON library counter functions. The counter size is 32 bit (4 byte) All counters are related to the PLOAM transmission process in upstream direction.

Note: This managed entity is not part of the reference implementation.

Managed Entity ID

Through an identical ID, this managed entity is implicitly linked to an instance of the related **ANI-G** managed entity.

Interval End Time

1 byte, read-only

This attribute identifies the most recently finished 15-minute interval and is implemented as in all other PMHD managed entities.

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The managed entities that are pointed to by this value provide alarm thresholds which are used for the monitored counters.

PLOAM Message Count

4 bytes, read-only

This attribute counts PLOAM messages transmitted upstream, excluding acknowledge messages. The value reported here is the sum of all counters *except for the number of acknowledge messages* as provided by the PON adapter (`pa_management_us_cnt, all_us`).

Serial Number ONU Message Count

4 bytes, read-only

This attribute counts Serial_number_ONU PLOAM messages transmitted. The value is provided by the PON adapter (`pa_management_us_cnt, serial_number`).

Registration Message Count

4 bytes, read-only

This attribute counts registration PLOAM messages transmitted. The value is provided by the PON adapter (`pa_management_us_cnt, registration`).

Key Report Message Count

4 bytes, read-only

This attribute counts key_report PLOAM messages transmitted. The value is provided by the PON adapter (`pa_management_us_cnt, key_report`).

Acknowledge Message Count

4 bytes, read-only

This attribute counts acknowledge PLOAM messages transmitted. It includes all forms of acknowledgment, including those transmitted in response to a PLOAM grant when the ONU has nothing to send. The value is provided by the PON adapter (`pa_management_us_cnt, acknowledge`).

Sleep Request Message Count

4 bytes, read-only

This attribute counts sleep_request PLOAM messages transmitted. The value is provided by the PON adapter (`pa_management_us_cnt, sleep_request`).

Actions

Create, delete, get, set, get current data

Attribute Value Change Notifications

There are no Attribute Value Changes defined for this managed entity.

Alarm Notifications

There are no Alarms defined for this managed entity.

4.20.10 XGS-PON Enhancements

This chapter describes managed entities which are defined only for XGS-PON implementations. These managed entities are not created if the system is run in ITU-T G.984 or G.987 operation mode.

4.20.10.1 Enhanced FEC Performance Monitoring History Data

ME class: 453

This managed entity is created at the request of the OLT and represents a single set of counters that accumulates the number of downstream FEC-related events for XGS-PON systems and newer. If downstream FEC mode is disabled, all counters deliver a value of 0.

The hardware counters are read regularly and accumulated in a set of 64-bit software counter variables. To read the counter values in XGS-PON operation mode use **fapi_pon_fec_counters_get.** To read the counter values in NG-PON2 operation mode use **fapi_pon_twdm_fec_counters_get.**

Note: This managed entity is not part of the reference implementation.

Readout of FEC Operation Mode

The downstream FEC operation mode is selected by the OLT through the PON hardware protocol. The FEC operation can be read through `fapi_pon_gpon_status_get` (`fec_status_ds`).

Managed Entity ID

2 bytes, read-only, set-by-create

In G-PON, XG-PON, and XGS-PON operation mode this is the same number as used for the **ANI-G**, to which this managed entity is related. In NG-PON2 operation mode this is the same number as used for the related **TWDM Channel** managed entity.

Note: In NG-PON2 operation mode, this managed entity cannot be connected to the ANI-G.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** managed entity that is pointed to by this value provides three alarm thresholds which are used for the Corrected Bytes, Corrected Code Words, Uncorrectable Code Words, and FEC Seconds. The **Threshold Data 2** managed entity is not used.

Corrected Bytes

8 bytes, read-only

Corrected bytes are counted by a hardware counter in the PON IP module and accumulated in a software counter by using the PON adapter function `pa_fec_pmhd_ops (cnt_corrected_bytes)`.

Corrected Code Words

8 bytes, read-only

Corrected code words are counted by a hardware counter in the PON IP module and accumulated in a software counter by using the PON adapter function `pa_fec_pmhd_ops (cnt_corrected_code_words)`.

Uncorrectable Code Words

8 bytes, read-only

Uncorrectable code words are counted by a hardware counter in the PON IP module and accumulated in a software counter by using the PON adapter function `pa_fec_pmhd_ops (cnt_uncorrected_code_words)`.

Total Code Words

8 bytes, read-only

The total number of received FEC code words (FEC blocks) is counted by a hardware counter in the PON IP module and accumulated in a software counter by using the PON adapter function `pa_fec_pmhd_ops (cnt_total_code_words)`.

FEC Seconds

2 bytes, read-only

This counts the number of one-second intervals in which at least one uncorrectable FEC error was seen.

Within the 15-minute time interval, a maximum increment of 900 can be seen.
Check `pa_fec_pmhd_ops (cnt_fec_seconds)` to retrieve the value.

Actions

Create, delete, get, set

Attribute Value Change Notifications

There are no Attribute Value Changes defined for this managed entity.

Alarm Notifications

The following notifications (threshold crossing alerts) are sent by this managed entity:

- Alarm #0: Corrected bytes
- Alarm #1: Corrected code words
- Alarm #2: Uncorrectable code words
- Alarm #4: FEC seconds

4.20.10.2 Enhanced TC Performance Monitoring History Data

ME class: 454

This managed entity is created at the request of the OLT. It is available in XGX-PON operation mode only and is not used on G-PON or XG-PON mode.

The error counters provided by this managed entity should be implemented by using the function as defined by `fapi_pon_xgtc_counters_get`.

Managed Entity ID

Through an identical ID, this managed entity is implicitly linked to an instance of the related **ANI-G** managed entity.

Interval End Time

1 byte, read-only

This attribute identifies the most recently finished 15-minute interval and is implemented as in all other PMHD managed entities.

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 64 bit ID

2 bytes, read, write, set-by-create

The **Threshold Data 64-bit** managed entity that is pointed to by this value provides the alarm thresholds.

PSBd HEC Error Count

4 bytes, read-only

The number of PSB downstream HEC errors in any of the fields of the downstream physical synchronization block is reported by the PON adapter (`pa_enhanced_tc_counters, psbd_hec_err_uncorr`).

XGTC HEC Error Count

4 bytes, read-only

The number of downstream XGTC HEC errors is reported by the PON adapter (`pa_enhanced_tc_counters, fs_hec_err_uncorr`).

Unknown Profile Count

4 bytes, read-only

This attribute counts the number of grants received whose specified profile was not known to the ONU.

The number of unknown profiles received is reported by the PON adapter (`pa_enhanced_tc_counters, burst_profile_err`).

Transmitted XGEM Frame Count

4 bytes, read-only

This attribute counts the number of non-idle XGEM frames transmitted. If an SDU is fragmented, each fragment is an XGEM frame and is counted as such.

The number of transmitted XGEM frames is reported by the PON adapter (`pa_enhanced_tc_counters, tx_frames`).

Fragment XGEM Frame Count

4 bytes, read-only

This attribute counts the number of XGEM frames that represent fragmented SDUs, as indicated by the LF bit = 0.

The number of transmitted XGEM frame fragments is reported by the PON adapter (`pa_enhanced_tc_counters, tx_fragments`).

XGEM HEC Lost Words Count

4 bytes, read-only

This attribute counts the number of four-byte words lost because of an XGEM frame HEC error. In general, all XGTC payload following the error is lost, until the next PSBd event.

The number of lost words is reported by the PON adapter (`pa_enhanced_tc_counters, lost_words`).

XGEM Key Error Count

4 bytes, read-only

This attribute counts the number of downstream XGEM frames received with an invalid key specification. The key may be invalid for several reasons.

The number of errors is reported by the PON adapter (`pa_enhanced_tc_counters, key_errors`).

XGEM HEC Error Count

4 bytes, read-only

This attribute counts the number of instances of an XGEM frame HEC error.

The number of errors is reported by the PON adapter (`pa_enhanced_tc_counters, xgem_hec_err_uncorr`).

Transmitted Bytes in Non-idle XGEM Frames Count

8 bytes, read-only

This attribute counts the number of bytes transmitted in non-idle XGEM upstream frames.

The number of errors is reported by the PON adapter (`pa_enhanced_tc_counters, tx_bytes`).

Received Bytes in Non-idle XGEM Frames Count

8 bytes, read-only

This attribute counts the number of bytes received in non-idle XGEM downstream frames.

The number of errors is reported by the PON adapter (`pa_enhanced_tc_counters, rx_bytes`).
The counter values are accumulated over all active XGEM ports.

LODS Event Count

4 bytes, read-only

This attribute counts the number of LODS events (PLOAM activation state transitions from O5.1 to O6).

The number of errors is reported by the PON adapter (`pa_enhanced_tc_counters, lods`).

LODS Event Restored Count

4 bytes, read-only

This attribute counts the number of restored (cleared) LODS events.

The number of errors is reported by the PON adapter (`pa_enhanced_tc_counters, lods_rest`).

LODS Event ONU Reactivation Count

4 bytes, read-only

This attribute counts the number of LODS events resulting in ONU reactivation without synchronization being reacquired.

The number of errors is reported by the PON adapter (`pa_enhanced_tc_counters, lods_react`).

Actions

Create, delete, get, set, get current data

Attribute Value Change Notifications

There are no Attribute Value Changes defined for this managed entity.

Alarm Notifications

The following notifications are sent by this managed entity:

- Alarm #1: PSBd HEC error count (triggered by threshold attribute #1)
- Alarm #2: XGTC HEC error count (triggered by threshold attribute #2)
- Alarm #3: Unknown profile count (triggered by threshold attribute #3)
- Alarm #4: XGEM HEC loss count (triggered by threshold attribute #4)
- Alarm #5: XGEM key errors (triggered by threshold attribute #5)
- Alarm #6: XGEM HEC error count (triggered by threshold attribute #6)

4.20.11 TWDM Enhancements

This section describes managed entities which are defined only for NG-PON2 and other TWDM implementations. These managed entities are not created when the system is run in an operation mode other than ITU-T G.989.

4.20.11.1 TWDM System Profile

ME class: 442

This managed entity is created autonomously by the ONU. It is available in NG-PON2 operation mode only and is not used in other ITU PON operation modes. It models the TWDM subsystem of the NG-PON2 system. An instance of this managed entity corresponds to a physical or virtual slot of the ONU, housing one or more access network interfaces (ANI-G).

Managed Entity ID

2 bytes, read-only

The managed entity ID is composed as a two-byte number, represented as 0xSS00. SS indicates the slot ID as defined by the **ANI-G** managed entity. For an integrated ONU with a single optical interface, the slot ID is 0 and so is the resulting managed entity ID.

Total TWDM Channel Number

1 byte, read-only

This attribute indicates the number of distinct TWDM channel pairs that the ONU supports in the given slot. The value provided depends on the optical interface which may be pluggable.

The value is pre-configured as the `channel_count` UCI configuration in `/etc/config/optic`.

The optical interface type is read through the DDMI two-wire interface upon first boot and the number of wavelength channels is taken from the related optical PMD database file.

The OMCI can read the value through the pon_lib function `fapi_pon_twdm_status_get(ch_count)`.

Channel Partition Index

1 byte, read, write

An operator may subdivide the set of TWDM and/or PtP WDM channels in an NG-PON2 system into non-overlapping subsets using an arbitrary criterion, such as commonality of service profile, equipment, or geographical location. Each such channel subset is known as a channel partition and is identified by an index which is unique within the NG-PON2 system. The Channel Partition Index is received in the Channel_Profile PLOAM message.

As an operational attribute, an ONU carries a Channel Partition Index, storing it in a non-volatile memory which is loaded upon OMCI start-up. An ONU's channel partition index can be specific (non-zero) or default (zero). An ONU with a specific channel partition index may activate and operate only on channels with a channel partition index that matches that of the ONU. An ONU with a default channel partition index can attempt to activate on a channel belonging to any channel partition, and learns its specific channel partition association at the time of ONU ID assignment.

An ONU with a specific channel partition index refuses an instruction to tune to a channel belonging to a different channel partition than that of the ONU.

The channel partition index value is represented as a 4-bit value, and maintained as a part of the OMCI MIB, rather than a TC layer config parameter. Values from 0 to 15 are valid, the attempt to set an invalid value must be rejected. The value 0 represents the default channel partition index, allowing the ONU to join any wavelength partition. Any other value restricts the ONU to wavelengths which belong to this channel partition. The channel partition of a wavelength is communicated in the channel profile PLOAM downstream message.

The configured value must be stored in non-volatile memory as part of the OMCI MIB initialization and be recovered upon system startup to the last known value. The value is configured to the PON IP by using `fapi_pon_twdm_cfg_set`.

This value is handed over to the PLOAM activation state machine upon start-up to restrict the wavelength usage to only those wavelengths which belong to the given CPI.

The handling is similar to that of the PLOAM state O7, which is also received from the PON firmware, stored in non-volatile memory, and used for subsequent system initialization.

The value is stored in the UCI configuration and can be read by using the pon_lib function `fapi_pon_twdm_cfg_get(ch_partition_index)` and updated by calling the related function `api_pon_twdm_cfg_set(ch_partition_index)`.

Channel Partition Waiver Timer

4 bytes, read, write

This attribute is an unsigned integer representing the time interval measured in seconds for which the ONU is blocked from an activation attempt on Channel Partition restriction. The timer measures the elapsed time from the moment an ONU finds a downstream wavelength channel belonging to a non-matching channel partition, to the moment the ONU resets its CPI in non-volatile memory to the default value (zero) in order to waive the CPI restriction. An alarm is raised by the underlying PON library when the timer runs out. This triggers the MIB change accordingly.

The default time is read from the UCI configuration.

Use `fapi_pon_gpon_cfg_set` (`ploam_timeout_cpi`) to initially configure this value from the UCI.

Upon a change by the OLT during operation, the new value is assigned through `fapi_pon_timeout_cfg_set (ploam_timeout_cpi)`.

The value of `0xFFFF` received from the OLT indicates infinity (no channel partition Waiver is granted), a value of 0 must be used in the call to `fapi_pon_timeout_cfg_set`.

LODS Re-initialization Timer

4 bytes, read-only

This attribute, which corresponds to timer TO2 of ITU-T G.989.3 expressed as an integer number of 125-µs PHY frame intervals, specifies the duration of time an ONU without configured wavelength channel protection (WLCP = OFF) waits in the Intermittent LODS (O6) state before transitioning into the Initial (O1) state for reactivation. The default value upon instantiation is 1000 (125 ms).

Use `fapi_pon_timeout_cfg_get` (`ploam_timeout_2`) to read the initial values from the pon_lib.

Use `fapi_pon_timeout_cfg_set` (`ploam_timeout_2`) to configure this value in steps of 1 ms. The pon_lib function will update the configuration in UCI.

LODS Protection Timer

4 bytes, read-only

This attribute, which corresponds to timer TO3 of ITU-T G.989.3 expressed as an integer number of 125-µs PHY frame intervals, specifies the duration of time an ONU with configured wavelength channel protection (WLCP = ON) waits in the Intermittent LODS (O6) state before transitioning into the Downstream tuning (O8) state to tune into the pre-configured protection wavelength channel. The default value upon instantiation is 200 (25 ms).

Use `fapi_pon_timeout_cfg_get` (`ploam_timeout_3`) to read the initial values from the pon_lib.

Use `fapi_pon_timeout_cfg_set` (`ploam_timeout_3`) to configure this value in steps of 1 ms. The pon_lib function will update the configuration in UCI.

Downstream Tuning Timer

4 bytes, read-only

This attribute, which corresponds to timer TO4 of ITU-T G.989.3 expressed as an integer number of 125-µs PHY frame intervals, specifies the duration of time an ONU in the downstream tuning (O8) state attempts to validate the specified target downstream wavelength channel (obtaining DWLCH ok to work), before transitioning into the Initial (O1) state for reactivation. Note that the Rx tuning time is properly included into this interval. The default value upon instantiation is 1000 (125 ms).

Use `fapi_pon_timeout_cfg_get` (`ploam_timeout_4`) to read the initial values from the pon_lib.

Use `fapi_pon_timeout_cfg_set` (`ploam_timeout_4`) to configure this value in steps of 1 ms. The pon_lib function will update the configuration in UCI.

Upstream Tuning Timer

4 bytes, read-only

This attribute, which corresponds to timer TO5 of ITU-T G.989.3 expressed as an integer number of 125-µs PHY frame intervals, specifies the duration of time an ONU in the Upstream tuning (O9) state attempts to obtain the upstream tuning confirmation in the specified target upstream wavelength channel before transitioning into the Initial (O1) state for reactivation. The default value upon instantiation is 1000 (125 ms).

Use `fapi_pon_timeout_cfg_get` (`ploam_timeout_5`) to read the initial values from the pon_lib.

Use `fapi_pon_timeout_cfg_set` (`ploam_timeout_5`) to configure this value in steps of 1 ms. The pon_lib function will update the configuration in UCI.

Location Label 1

24 bytes, read, write

This attribute represents the first part of the field, which is written by the OLT to provide the topological location information for the specific OLT channel termination within the operator domain. This attribute is not interpreted by the ONU, but may be used by a dual-managed ONU as a part of an alarm report provided over non-OMCI management channel.

The value is stored in the OMCI database to be readable by other processes or dumped for debug purpose.

Location Label 2

24 bytes, read, write

This attribute represents the second part of the field, which is written by the OLT to provide the topological location information for the specific OLT channel termination within the operator domain. This attribute is not interpreted by the ONU, but may be used by a dual-managed ONU as a part of an alarm report provided over non-OMCI management channel.

The value is stored in the OMCI database to be readable by other processes or dumped for debug purpose.

Actions

Get, set

Attribute Value Change Notifications

There are no attribute value changes defined for this managed entity.

Alarm Notifications

There are no alarms defined for this managed entity.

Default Timeout Values

The default values of the timeout attributes are configured in the UCI file `/etc/config/gpon`.
See **Chapter 2.4.3.4** for more detail.

4.20.11.2 TWDM Channel

ME class: 443

This managed entity is created by the ONU autonomously. It is available in NG-PON2 operation mode only and is not used in single-wavelength ITU PON operation modes.

Managed Entity ID

2 bytes, read-only

This two-byte number is represented as 0xSS0C, where SS indicates the ONU slot ID as used by the **TWDM System Profile**, and C is the TWDM channel managed entity number assigned by the ONU itself, starting from 0 in ascending order up to 3.

Active Channel Indication

1 byte, read-only

This attribute indicates whether the managed entity is associated with an active TWDM channel, in which the ONU can receive a downstream optical signal and transmit upstream as instructed. The default value is FALSE. The ONU sets the attribute to TRUE after it confirmed the channel is active.

This is indicated by the reception of the `Channel_Profile` PLOAM message. Once this message is initially received or the profile has changed to a new version, the PON firmware sends the TWDM_CHANNEL_PROFILE message event. This message provides the information about which wavelength channels are active in the system.

From OMCI perspective all TWDM channel MEs associated with a given slot/circuit pack *can* be active at the same time.

The ONU reverts the attribute to FALSE when the OLT withdraws a channel through the `Channel_Profile` PLOAM messages for that channel, by setting the channel `void` for the upstream and/or downstream direction.

Use `fapi_pon_twdm_channel_profile_status_get` function to read the information about the received channel profiles. There are up to 16 channel profiles possible, addressed by their `cp_id` value (0 to 15).

If one of the profiles matches the upstream and downstream channel index values, this attribute is set TRUE (1). Otherwise it is set FALSE (0).

Operational Channel Indication

1 byte, read-only

This attribute is set to TRUE for an active TWDM channel in which the ONT is currently operating.

The operational statistics is accumulated in the PMHD managed entities associated with that TWDM channel.

Use `fapi_pon_twdm_status_get` to read the information about the currently operational wavelength channel. This value is set to TRUE when the reported active channels match the downstream and upstream wavelengths defined by the **Downstream Wavelength Channel Index** and **Upstream Wavelength Channel Index** attributes.

Downstream Wavelength Channel Index

1 byte, read-only

This attribute identifies the downstream wavelength channel in reference to Table 69. For a fixed wavelength TDM system, the downstream wavelength channel number is 0x0F. For an inactive channel it has value 0xFF.This value is preset in the MIB initialization file.

Table 69 Downstream Wavelength Channel Index Assignment

Channel Index	Frequency (THz)	Wavelength (nm)
1	187.8	1596.34
2	187.7	1597.19
3	187.6	1598.04
4	187.5	1598.89
5	187.4	1599.75
6	187.3	1600.60
7	187.2	1601.46
8	187.1	1602.31

Upstream Wavelength Channel Index

1 byte, read-only

For an active TWDM channel, this attribute identifies the upstream wavelength channel in reference to Table 70. For a fixed wavelength TDM system, the downstream wavelength channel number is 0x0F. For an inactive channel its value of 0xFF.

This value is defined in the MIB initialization file starting at 1 and incrementing for each supported wavelength channel.

Table 70 Upstream Wavelength Channel Index Assignment

Channel Index	50 GHz Channel Spacing		100 GHz Channel Spacing		200 GHz Channel Spacing	
	Frequency (THz)	Wavelength (nm)	Frequency (THz)	Wavelength (nm)	Frequency (THz)	Wavelength (nm)
1	195.25	1535.43	195.6	1532.68	196.1	1528.77
2	195.20	1535.82	195.5	1533.47	195.9	1530.33
3	195.15	1536.22	195.4	1534.25	195.7	1531.90
4	195.10	1536.61	195.3	1535.04	195.5	1533.47
5	195.05	1537.00	195.2	1535.82	195.3	1535.04
6	195.00	1537.40	195.1	1536.61	195.1	1536.61
7	194.95	1537.79	195.0	1537.40	194.9	1538.19
8	194.90	1538.19	194.9	1538.19	194.7	1538.77

Actions

Get, set

Attribute Value Change Notifications

There are no Attribute Value Changes defined for this managed entity.

Alarm Notifications

There are no Alarms defined for this managed entity.

MIB Initialization File Example

This example handles the presence of four wavelength channel pairs. The ONU slot ID is assumed as 0x80.

```
## TWDM Channel #0
443 0x8000 0 0 0xFF 0xFF

## TWDM Channel #1
443 0x8001 0 0 0xFF 0xFF

## TWDM Channel #2
443 0x8002 0 0 0xFF 0xFF

## TWDM Channel #3
443 0x8003 0 0 0xFF 0xFF
```

4.20.11.3 TWDM Channel PHY/LODS Performance Monitoring History Data

ME class: 444

This managed entity collects certain performance monitoring data associated with the slot/circuit pack, hosting one or more ANI-G MEs, and a specific TWDM channel. Instances of this managed entity are created and deleted by the OLT.

An instance of this managed entity is associated with an instance of TWDM channel managed entity. It is available in NG-PON2 operation mode only and is not used in other ITU PON operation modes.

Managed Entity ID

2 bytes, set-by-create, read-only

Through an identical ID, this managed entity is implicitly linked to an instance of the TWDM channel managed entity

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** and **Threshold Data 2** managed entities that are pointed to by this value provide alarm thresholds which are used for the monitoring items.

Number of BIP-32 Protected Words

8 bytes, read-only

This counter accumulates the number of 4-byte words included in BIP-32 check. This is a product of the number of downstream FS frames received by the size of the downstream FS frame after the FEC parity byte, if any, have been removed. The count applies to the entire downstream data flow, whether or not addressed to that ONU.

Use fapi_pon_gtc_counters_get to read the counter value.

BIP-32 Error Count

4 bytes, read-only

This counter accumulates the number of bit errors in the received downstream FS frames as measured using BIP-32. If FEC is supported in the downstream direction, the BIP-32 count applies to the downstream FS frame after the FEC correction has been applied and the FEC parity bytes have been removed.

Use fapi_pon_fec_counters_get to read the counter value.

Corrected Downstream PSBd HEC Error Count

4 bytes, read-only

This counter accumulates the number of errors in either CFC or OCS fields of the PSBd block which *have been corrected* using the HEC error correction method.

Use fapi_pon_xgtc_counters_get to read the counter value.

Uncorrectable Downstream PSBd HEC Error Count

4 bytes, read-only

This counter accumulates the number of errors in either CFC or OCS fields of the PSBd block which *could not be corrected* using the HEC error correction method.

Use fapi_pon_xgtc_counters_get to read the counter value.

Corrected Downstream FS Header HEC Error Count

4 bytes, read-only

This counter accumulates the number of errors in the downstream FS header which *have been corrected* using the HEC error correction method.

Uncorrectable Downstream FS Header HEC Error Count

4 bytes, read-only

This counter accumulates the number of errors in the downstream FS header which *could not be corrected* using the HEC error correction method.

Total Number of LODS Events

4 bytes, read-only

This counter accumulates the number of state transitions from O5.1/O5.2 to O6, referring to the ONU activation cycle state machine.

LODS Events Restored in the Operating TWDM Channel

4 bytes, read-only

This counter accumulates the number of LODS events cleared automatically without retuning.

LODS Events Restored in the Protection TWDM Channel

4 bytes, read-only

This counter accumulates the number of events resolved by retuning to a pre-configured protection TWDM channel. The event is counted against the original operating channel.

LODS Events Restored in a Discretionary TWDM Channel

4 bytes, read-only

This counter accumulates the number of LODS events resolved by retuning to a TWDM channel chosen by the ONU. Implies that the wavelength channel protection for the operating channel is not active. The event is counted against the original operating channel.

LODS Events Resulting in Reactivation

4 bytes, read-only

This counter accumulates the number of LODS events resolved through ONU reactivation. That is either TO2 (without WLCP) or TO3+TO4 (with WLCP) expire before the downstream channel is reacquired, referring to the ONU activation cycle state machine. The event is counted against the original operating channel.

LODS Events Resulting in Reactivation after Retuning to the Protection TWDM Channel

4 bytes, read-only

This counter accumulates the number of LODS events resolved through ONU reactivation after attempted protection switching, which turns out to be unsuccessful due to a handshake failure.

LODS Events Resulting in Reactivation after Retuning to the Discretionary TWDM Channel

4 bytes, read-only

This counter accumulates the number of LODS events resolved through ONU reactivation after attempted switching to a discretionary TWDM channel, which turns out to be unsuccessful due to a handshake failure.

Actions

Create, delete, get, set

Attribute Value Change Notifications

There are no Attribute Value Changes defined for this managed entity.

Alarm Notifications

The following notifications (threshold crossing alerts) are sent by this managed entity.

- Alarm #0: N/A
- Alarm #1: BIP-32 error count
- Alarm #2: Corrected PSBd HEC errors
- Alarm #3: Uncorrectable PSBd HEC errors
- Alarm #4: Corrected FS header errors
- Alarm #5: Uncorrectable FS header errors
- Alarm #6: Total LODS event count
- Alarm #7: LODS restored in operating TWDM channel
- Alarm #8: LODS restored in protection TWDM channel
- Alarm #9: LODS restored in discretionary TWDM channel
- Alarm #10: LODS reactivations
- Alarm #11: LODS handshake failure in protection channel
- Alarm #12: LODS handshake failure in discretionary channel

4.20.11.4 TWDM Channel XGEM Performance Monitoring History Data

ME class: 445

This managed entity is created and deleted by the OLT. It is available in NG-PON2 operation mode only and is not used in other ITU PON operation modes. It is related to an instance of the **TWDM Channel** managed entity.

Managed Entity ID

2 bytes, read-only, set-by-create

The same number is used as for the **TWDM Channel**, to which this managed entity is related.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data

2 bytes, read, write, set-by-create

The **Threshold Data 64-bit** managed entity that is pointed to by this value provides alarm thresholds which are used to trigger the TCA alarms.

Total Transmitted XGEM Frames

8 bytes, read-only

Accumulated number of XGEM frames which have been transmitted over all XGEM ports belonging to this ONU.

Use `fapi_pon_gem_port_counters_get` to read the counter value `tx_frames` for each XGEM port which is active on the ONU. This includes the XGEM port used for the OMCC.

Transmitted XGEM Frames with LF Bit not set

8 bytes, read-only

Accumulated number of XGEM frames with LF == 0 which have been transmitted on any XGEM port belonging to this ONU.

Use `fapi_pon_gem_port_counters_get` to read the counter value `tx_fragments` for each XGEM port which is active on the ONU. This includes the XGEM port used for the OMCC.

Total Received XGEM Frames

8 bytes, read-only

Accumulated number of XGEM frames which have been received on any XGEM port belonging to this ONU.

Use `fapi_pon_gem_port_counters_get` to read the counter value `rx_frames` for each XGEM port which is active on the ONU. This includes the XGEM port used for the OMCC.

Received XGEM Frames with XGEM Header HEC Errors

8 bytes, read-only

Accumulated number of XGEM frame delineation loss events caused by XGEM HEC errors.

Use `fapi_pon_gtc_counters_get` to read the counter value `dic_gem_frames`.

FS Words Lost due to XGEM HEC Errors

8 bytes, read-only

Accumulated number of FS frame words lost due to XGEM frame delineation loss events caused by XGEM HEC errors.

Note: Implementation is t.b.d.

XGEM Encryption Key Errors

8 bytes, read-only

The counter aggregated across all XGEM ports of the given ONU identifies the number of received XGEM frames that must be discarded because of an unknown or invalid encryption key. This number is included in the **Total Received XGEM Frames** count.

Use `fapi_pon_gem_port_counters_get` to read the counter value `key_errors` for each XGEM port which is active on the ONU. This includes the XGEM port used for the OMCC.

Transmitted Bytes in non-idle XGEM Frames

8 bytes, read-only

Accumulated number of bytes transmitted in non-idle frames on any XGEM port belonging to this ONU.

Use `fapi_pon_gem_port_counters_get` to read the counter value `tx_bytes` for each XGEM port which is active on the ONU. This includes the XGEM port used for the OMCC.

Received Bytes in non-idle XGEM Frames

8 bytes, read-only

Accumulated number of bytes received in non-idle frames on any XGEM port belonging to this ONU.

Use `fapi_pon_gem_port_counters_get` to read the counter value `rx_bytes` for each XGEM port active on the ONU. This includes the XGEM port used for the OMCC.

Actions

Create, delete, get, get current data, set

Alarm Notifications

The following notifications (threshold crossing alerts) are sent by this managed entity.

- Alarm #0: Total transmitted XGEM frames
- Alarm #1: Transmitted XGEM frames with LF == 0
- Alarm #2: Total received XGEM frames
- Alarm #4: FS words lost due to XGEM HEC errors
- Alarm #5: XGEM encryption key errors

4.20.11.5 TWDM Channel PLOAM Performance Monitoring History Data Part 1

ME class: 446

This managed entity is created and deleted by the OLT. It is available in NG-PON2 operation mode only and is not used in other ITU PON operation modes.

The downstream PLOAM message counts of this manage entity include only the received PLOAM messages that are targeted at the ONU, which are:

- Unicast PLOAM messages, addressed by ONU-ID
- Broadcast PLOAM messages, addressed by the ONU's serial number
- Broadcast PLOAM messages, addressed to all ONUs on the PON

Managed Entity ID

2 bytes, read-only, set-by-create

The same number is used as for the **TWDM Channel**, to which this managed entity is related.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the "synchronize time" action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** and **Threshold Data 2** managed entities that are pointed to by this value provide alarm thresholds which are used for the monitoring items.

PLOAM MIC Errors

4 bytes, read-only

This attribute provides the number of received PLOAM messages that were dropped due to an invalid MIC value.

Downstream PLOAM Messages

4 bytes, read-only

This attribute provides the total number of received PLOAM messages with a valid MIC field.

Ranging Time PLOAM Messages

4 bytes, read-only

This attribute provides the number of received "Ranging Time" PLOAM messages.

Protection Control PLOAM Messages

4 bytes, read-only

This attribute provides the number of received "Protection Control" PLOAM messages.

Adjust Tx Wavelength PLOAM Messages

4 bytes, read-only

This attribute provides the number of received "Adjust Tx Wavelength" PLOAM messages.

Adjust Tx Wavelength Adjustment Amplitude PLOAM Messages

4 bytes, read-only

This attribute provides the number of received “Adjust Tx Wavelength Adjustment Amplitude” PLOAM messages.

Actions

Create, delete, get, get current data, set

Notifications

The following notification is sent by this managed entity.

- Alarm #0: PLOAM MIC Errors

4.20.11.6 TWDM Channel PLOAM Performance Monitoring History Data Part 2

ME class: 447

This managed entity is created and deleted by the OLT. It is available in NG-PON2 operation mode only and is not used in other ITU PON operation modes.

The downstream PLOAM message counts of this manage entity include only the received PLOAM messages that are targeted at the ONU, which are:

- Unicast PLOAM messages, addressed by ONU-ID
- Broadcast PLOAM messages, addressed by the ONU's serial number
- Broadcast PLOAM messages, addressed to all ONUs on the PON

Managed Entity ID

2 bytes, read-only, set-by-create

The same number is used as for the **TWDM Channel**, to which this managed entity is related.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the "synchronize time" action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** and **Threshold Data 2** managed entities that are pointed to by this value provide alarm thresholds which are used for the monitoring items.

System Profile PLOAM Messages

4 bytes, read-only

This attribute provides the number of received "System Profile" PLOAM messages.

Channel Profile PLOAM Messages

4 bytes, read-only

This attribute provides the number of received "Channel Profile" PLOAM messages.

Burst Profile PLOAM Messages

4 bytes, read-only

This attribute provides the number of received "Burst Profile" PLOAM messages.

Assign ONU ID PLOAM Messages

4 bytes, read-only

This attribute provides the number of received "ONU ID" PLOAM messages.

Unsatisfied Adjust Tx Wavelength Requests

4 bytes, read-only

This attribute provides the number of received "Tx Wavelength Request" PLOAM messages.

Deactivate ONU ID PLOAM Messages

4 bytes, read-only

This attribute provides the number of received “Deactivate ONU ID” PLOAM messages.

Disable Serial Number PLOAM Messages

4 bytes, read-only

This attribute provides the number of received “Disable Serial Number” PLOAM messages.

Request Registration PLOAM Messages

4 bytes, read-only

This attribute provides the number of received “Registration” PLOAM messages.

Assign Alloc ID PLOAM Messages

4 bytes, read-only

This attribute provides the number of received “Assign Alloc ID” PLOAM messages.

Key Control PLOAM Messages

4 bytes, read-only

This attribute provides the number of received “Key Control” PLOAM messages.

Sleep Allow PLOAM Messages

4 bytes, read-only

This attribute provides the number of received “Sleep Allow” PLOAM messages.

Tuning Control/Request PLOAM Messages

4 bytes, read-only

This attribute provides the number of received “Tuning Control/Request” PLOAM messages.

Tuning Control/Complete_d PLOAM Messages

4 bytes, read-only

This attribute provides the number of received “Tuning Control/Complete_d” PLOAM messages.

Calibration Request PLOAM Messages

4 bytes, read-only

This attribute provides the number of received “Calibration Request” PLOAM messages.

Actions

Create, delete, get, get current data, set

Notifications

The following notification is sent by this managed entity.

- Alarm #0: Unsatisfied Adjust Tx Wavelength Requests

4.20.11.7 TWDM Channel PLOAM Performance Monitoring History Data Part 3

ME class: 448

This managed entity is created and deleted by the OLT. It is available in NG-PON2 operation mode only and is not used in other ITU PON operation modes.

This managed entity provides counters for the upstream PLOAM messages.

Managed Entity ID

2 bytes, read-only, set-by-create

The same number is used as for the **TWDM Channel**, to which this managed entity is related.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** and **Threshold Data 2** managed entities that are pointed to by this value provide alarm thresholds which are used for the monitoring items.

Upstream PLOAM Messages

4 bytes, read-only

This attribute provides the total number of transmitted PLOAM messages.

The number of Acknowledge messages are not included in this value.

Serial Number ONU In-band PLOAM Messages

4 bytes, read-only

This attribute provides the number of transmitted in-band “Serial Number ONU” PLOAM messages.

Serial Number ONU AMCC PLOAM Messages

4 bytes, read-only

This attribute provides the number of transmitted AMCC “Serial Number ONU” PLOAM messages.

Note: This counter value is always reported as 0, because the AMCC function is not supported.

Registration PLOAM Messages

4 bytes, read-only

This attribute provides the number of transmitted “Registration” PLOAM messages.

Key Report PLOAM Messages

4 bytes, read-only

This attribute provides the number of transmitted “Key Report” PLOAM messages.

Acknowledgment PLOAM Messages

4 bytes, read-only

This attribute provides the number of transmitted “Acknowledgment” PLOAM messages.

Sleep Request PLOAM Messages

4 bytes, read-only

This attribute provides the number of transmitted “Sleep Request” PLOAM messages.

Tuning Response (ACK/NACK) PLOAM Messages

4 bytes, read-only

This attribute provides the number of transmitted “Tuning Response (ACK/NACK)” PLOAM messages.

Tuning Response (Complete_u/Rollback) PLOAM Messages

4 bytes, read-only

This attribute provides the number of transmitted “Tuning Response (Complete_u/Rollback)” PLOAM messages.

Power Consumption Report PLOAM Messages

4 bytes, read-only

This attribute provides the number of transmitted “Power Consumption Report” PLOAM messages.

Change Power Level Parameter Error PLOAM Messages

4 bytes, read-only

This attribute provides the number of transmitted Acknowledgment PLOAM messages with Parameter Error completion code in response to a Change_Power_Level PLOAM message.

Actions

Create, delete, get, get current data, set

Notifications

The following notification is sent by this managed entity.

- Alarm #0: Change Power Level parameter error count

4.20.11.8 TWDM Channel Tuning Performance Monitoring History Data Part 1

ME class: 449

This managed entity is created and deleted by the OLT. It is available in NG-PON2 operation mode only and is not used in other ITU PON operation modes.

This managed entity provides counters for the wavelength channel tuning process.

Managed Entity ID

2 bytes, read-only, set-by-create

The same number is used as for the **TWDM Channel**, to which this managed entity is related.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** and **Threshold Data 2** managed entities that are pointed to by this value provide alarm thresholds which are used for the monitoring items.

Tuning Control Requests for Rx only or Rx and Tx

4 bytes, read-only

The number of received Tuning Control PLOAM messages with Request operation code that contain tuning instructions either for the receiver only or for both the receiver and the transmitter.

Tuning Control Requests for Tx only

4 bytes, read-only

The number of received Tuning Control PLOAM messages with Request operation code that contain tuning instructions only for the transmitter.

Tuning Control Requests Rejected/INT_SFC

4 bytes, read-only

The number of transmitted Tuning Response PLOAM messages with NACK operation code and INT_SFC response code, indicating the inability to start transceiver tuning by the specified time (SFC).

Tuning Control Requests Rejected/DS_xxx

4 bytes, read-only

The aggregate number of transmitted Tuning Response PLOAM messages with NACK operation code and any DS_xxx response code, indicating target downstream wavelength channel inconsistency.

Tuning Control Requests Rejected/US_xxx

4 bytes, read-only

The aggregate number of transmitted Tuning Response PLOAM messages with NACK operation code and any US_xxx response code, indicating target upstream wavelength channel inconsistency.

Tuning Control Requests Fulfilled with ONU reacquired at target channel

4 bytes, read-only

The number of controlled tuning attempts for which an upstream tuning confirmation has been obtained in the target channel.

Tuning Control Requests Failed due to Target DS Wavelength Channel not Found

4 bytes, read-only

The number of controlled tuning attempts that failed due to timer expiration (TO4) in the DS Tuning state (O8) in the target channel.

Tuning Control Requests Failed due to no Feedback in Target DS Wavelength Channel

4 bytes, read-only

The number of controlled tuning attempts that failed due to timer expiration (TO5) in the US Tuning state (O9) in the target channel.

Tuning Control Requests Resolved with ONU Reacquired at the Discretionary Channel

4 bytes, read-only

The number of controlled tuning attempts for which an upstream tuning confirmation has been obtained in the discretionary channel..

Tuning Control Requests Rollback/COM_DS

4 bytes, read-only

The number of controlled tuning attempts that failed due to the communication condition in the target channel, as indicated by the Tuning Response PLOAM message with Rollback operation code and COM_DS response code.

Tuning Control Requests Rollback/DS_xxx

4 bytes, read-only

The aggregate number of controlled tuning attempts that failed due to target downstream wavelength channel inconsistency, as indicated by the Tuning Response PLOAM message with Rollback operation code and any DS_xxx response code.

Tuning Control Requests Rollback/US_xxx

4 bytes, read-only

The aggregate number of controlled tuning attempts that failed due to target upstream wavelength channel parameter violation, as indicated by the Tuning Response PLOAM message with Rollback operation code and US_xxx response code.

Tuning Control Requests Failed with ONU Reactivation

4 bytes, read-only

The number of controlled tuning attempts that failed due to any reason, with expiration of timers TO4 or TO5 causing the ONU transition into state O1.

Actions

Create, delete, get, get current data, set

Notifications

The following Threshold Crossing Alert notifications are sent by this managed entity.

- Alarm #0: Tuning control requests rejected/INT_SFC
- Alarm #1: Tuning control requests rejected/DS_xxx
- Alarm #2: Tuning control requests rejected/US_xxx
- Alarm #3: Tuning control requests failed because TO4 expired
- Alarm #4: Tuning control requests failed because TO5 expired
- Alarm #5: Tuning control requests Rollback/COM_DS
- Alarm #6: Tuning control requests Rollback/DS_xxx
- Alarm #7: Tuning control requests Rollback/US_xxx
- Alarm #8: Tuning control requests failed/Reactivation

4.20.11.9 TWDM Channel Tuning Performance Monitoring History Data Part 2

ME class: 450

This managed entity is created and deleted by the OLT. It is available in NG-PON2 operation mode only and is not used in other ITU PON operation modes.

This managed entity provides counters for the wavelength channel tuning process.

Managed Entity ID

2 bytes, read-only, set-by-create

The same number is used as for the **TWDM Channel**, to which this managed entity is related.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** and **Threshold Data 2** managed entities that are pointed to by this value provide alarm thresholds which are used for the monitoring items.

Tuning Control Requests Rejected/DS_ALBL

4 bytes, read-only

The number of transmitted Tuning Response PLOAM messages with NACK operation code and DS_ALBL response code, indicating downstream administrative label inconsistency.

Tuning Control Requests Rejected/DS_VOID

4 bytes, read-only

The number of transmitted Tuning Response PLOAM messages with NACK operation code and DS_VOID response code, indicating that the target downstream wavelength channel descriptor is void.

Tuning Control Requests Rejected/DS_PART

4 bytes, read-only

The number of transmitted Tuning Response PLOAM messages with NACK operation code and DS_PART response code, indicating that tuning request involves channel partition violation.

Tuning Control Requests Rejected/DS_TUNR

4 bytes, read-only

The number of transmitted Tuning Response PLOAM messages with NACK operation code and DS_TUNR response code, indicating that the target DS wavelength channel is out of receiver tuning range.

Tuning Control Requests Rejected/DS_LNRT

4 bytes, read-only

The number of transmitted Tuning Response PLOAM messages with NACK operation code and DS_LNRT response code, indicating downstream line rate inconsistency in the target channel.

Tuning Control Requests Rejected/DS_LNCD

4 bytes, read-only

The number of transmitted Tuning_Response PLOAM messages with NACK operation code and DS_LNCD response code, indicating downstream line code inconsistency in the target channel.

Tuning Control Requests Rejected/US_ALBL

4 bytes, read-only

The number of transmitted Tuning_Response PLOAM messages with NACK operation code and US_ALBL response code, indicating upstream administrative label inconsistency.

Tuning Control Requests Rejected/US_VOID

4 bytes, read-only

The number of transmitted Tuning Response PLOAM messages with NACK operation code and US_VOID response code, indicating that the target upstream wavelength channel descriptor is void.

Tuning control requests rejected/US_TUNR

4 bytes, read-only

The number of transmitted Tuning Response PLOAM messages with NACK operation code and US_TUNR response code, indicating that the target US wavelength channel is out of transmitter tuning range.

Tuning control requests rejected/US_CLBR

4 bytes, read-only

The number of transmitted Tuning Response PLOAM messages with NACK operation code and US_CLBR response code, indicating that the transmitter has insufficient calibration accuracy in the target US wavelength channel.

Tuning control requests rejected/US_LKTP

4 bytes, read-only

The number of transmitted Tuning Response PLOAM messages with NACK operation code and US_LKTP response code, indicating upstream optical link type inconsistency.

Tuning control requests rejected/US_LNRT

4 bytes, read-only

The number of transmitted Tuning _Response PLOAM messages with NACK operation code and US_LNRT response code, indicating upstream line rate inconsistency in the target channel.

Tuning control requests rejected/US_LNCD

4 bytes, read-only

The number of transmitted Tuning Response PLOAM messages with NACK operation code and US_LNCD response code, indicating upstream line code inconsistency in the target channel.

Actions

Create, delete, get, get current data, set

Notifications

The following notification is sent by this managed entity.

- Alarm #0: Tuning control requests rejected/DS_ALBL
- Alarm #1: Tuning control requests rejected/DS_VOID
- Alarm #2: Tuning control requests rejected/DS_PART
- Alarm #3: Tuning control requests rejected/DS_TUNR
- Alarm #4: Tuning control requests rejected/DS_LNRT
- Alarm #5: Tuning control requests rejected/DS_LNCD
- Alarm #6: Tuning control requests rejected/US_ALBL
- Alarm #7: Tuning control requests rejected/US_VOID
- Alarm #8: Tuning control requests rejected/US_TUNR
- Alarm #9: Tuning control requests rejected/US_CLBR
- Alarm #10: Tuning control requests rejected/US_LKTP
- Alarm #11: Tuning control requests rejected/US_LNRT
- Alarm #12: Tuning control requests rejected/US_LNCD

4.20.11.10TWDM Channel Tuning Performance Monitoring History Data Part 3

ME class: 451

This managed entity is created and deleted by the OLT. It is available in NG-PON2 operation mode only and is not used in other ITU PON operation modes.

This managed entity provides counters for the wavelength channel tuning process.

Managed Entity ID

2 bytes, read-only, set-by-create

The same number is used as for the **TWDM Channel**, to which this managed entity is related.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** and **Threshold Data 2** managed entities that are pointed to by this value provide alarm thresholds which are used for the monitoring items.

Tuning control requests Rollback/DS_ALBL

4 bytes, read-only

The number of controlled tuning attempts that failed due to downstream administrative label inconsistency, as indicated by the Tuning Response PLOAM message with Rollback operation code and DS_ALBL response code.

Tuning control requests Rollback/DS_LKPT

4 bytes, read-only

The number of controlled tuning attempts that failed due to downstream optical link type inconsistency, as indicated by the Tuning Response PLOAM message with Rollback operation code and DS_LKTP response code.

Tuning control requests Rollback/US_ALBL

4 bytes, read-only

The number of controlled tuning attempts that failed due to upstream administrative label violation, as indicated by the Tuning Response PLOAM message with Rollback operation code and US_ALBL response code.

Tuning control requests Rollback/US_VOID

4 bytes, read-only

The number of controlled tuning attempts that failed due to the target upstream wavelength channel descriptor being void, as indicated by the Tuning Response PLOAM message with Rollback operation code and US_VOID response code.

Tuning control requests Rollback/US_TUNR

4 bytes, read-only

The number of controlled tuning attempts that failed due to the transmitter tuning range violation, as indicated by the Tuning Response PLOAM message with Rollback operation code and US_TUNR response code.

Tuning control requests Rollback/US_LKTP

4 bytes, read-only

The number of controlled tuning attempts that failed due to the upstream optical link type violation, as indicated by the Tuning Response PLOAM message with Rollback operation code and US_LKTP response code..

Tuning control requests Rollback/US_LNRT

4 bytes, read-only

The number of controlled tuning attempts that failed due to the upstream line rate violation, as indicated by the Tuning Response PLOAM message with Rollback operation code and US_LNRT response code..

Tuning control requests Rollback/US_LNCD

4 bytes, read-only

The number of controlled tuning attempts that failed due to the upstream line code violation, as indicated by the Tuning Response PLOAM message with Rollback operation code and US_LNCD response code.

Actions

Create, delete, get, get current data, set

Notifications

The following notification is sent by this managed entity:

- Alarm #0: Tuning control requests Rollback/DS_ALBL
- Alarm #1: Tuning control requests Rollback/DS_LKTP
- Alarm #2: Tuning control requests Rollback/US_ALBL
- Alarm #3: Tuning control requests Rollback/US_VOID
- Alarm #4: Tuning control requests Rollback/US_PUNR
- Alarm #5: Tuning control requests Rollback/US_LKTP
- Alarm #6: Tuning control requests Rollback/US_LNRT
- Alarm #7: Tuning control requests Rollback/US_LNCD

4.20.11.11TWDM Channel OMCI Performance Monitoring History Data

ME class: 452

This managed entity is created and deleted by the OLT. It is available in NG-PON2 operation mode only and is not used in other ITU PON operation modes.

This managed entity provides counters for the wavelength channel tuning process.

Managed Entity ID

2 bytes, read-only, set-by-create

The same number is used as for the **TWDM Channel**, to which this managed entity is related.

Interval End Time

1 byte, read-only

The sequence of 15-minute intervals is initiated by the “synchronize time” action, issued by the OLT against the ONU-G managed entity. This establishes a 15-minute tick boundary and starts numbering the intervals from 0. This interval number is returned in this attribute.

Threshold Data 1/2

2 bytes, read, write, set-by-create

The **Threshold Data 1** and **Threshold Data 2** managed entities that are pointed to by this value provide alarm thresholds which are used for the monitoring items.

OMCI Baseline Message Count

4 bytes, read-only

This attribute provides the number of baseline format OMCI messages directed to the ONU on the related TWDM channel.

OMCI Extended Message Count

4 bytes, read-only

This attribute provides the number of extended format OMCI messages directed to the ONU on the related TWDM channel.

OMCI MIC Error Count

4 bytes, read-only

This attribute provides the number of OMCI messages received with a MIC error on the related TWDM channel.

Actions

Create, delete, get, get current data, set

Notifications

The following notification is sent by this managed entity.

- Alarm #0: OMCI MIC error count

4.20.12 Vendor-specific OMCI Enhancements for Verizon

For interoperability testing there are several vendor-specific managed entities defined by Verizon's "OpenOMCI" Specification **[39]**. These are the managed entities which are not included in the amendments or updated versions of ITU-T G.988.

These managed entities are needed only if the system must support Verizon OpenOMCI operation.

- **Verizon OpenOMCI**
- **Watchdog Configuration Data**
- **Watchdog PMHD**

4.20.12.1 Verizon OpenOMCI

There is a dedicated managed entity to identify the supported version of OpenOMCI to the OLT. This is required for Verizon inter-operation testing.

ME class: 65400

This managed entity is created by the ONU to provide its compliance capability.

Managed Entity ID

2 bytes, read-only

This value is configured in the default MIB initialization file.
There is only one instance of this managed entity with ID = 0.

Supported Specification Version

4 bytes, read-only

This attribute uniquely identifies the highest version of the Verizon OpenOMCI [39] specification that the ONU supports. The version is represented by a pair of integer values (R, V), where R, the two most significant bytes, is the major release, and V, the least significant bytes, is the minor release. For backward compatibility purposes the ONU is expected to support all lower numbered versions within the same release of the OpenOMCI specification, as well as the generic ITU-T G.988 specification without OpenOMCI features.

This value is configured in the default MIB initialization file.

The supported revision is 2.2 (0x0002 0002).

PON Device Type

2 bytes, read-only

This attribute refers to the ONU type specified by the operator in the appropriate requirement documents. Together with the supported specification version attribute, it unambiguously identifies the set of managed entities, attributes and features that are supported by the ONU. The following list provides the PON device type values supported in revision 2.2 of the specification.

- 0x0004: NG-PON2 SFU ONT (Ethernet + VoIP)
- 0x0006: NG-PON2 ETH ONT (Ethernet only)
- 0x0008: NG-PON2 SOHO ONT (Small office/home office)
- 0x000C: NG-PON2 IBONT (Integrated Business ONT)
- 0x000E: NG-PON2 BHR (Broadband Home Router)

Specification Version in Use

4 bytes, read, write

This attribute is populated by the OLT to indicate an agreement to use the **Verizon OpenOMCI** specification and its effective version. If the OLT does not access the **Verizon OpenOMCI** managed entity, or if it sets the value of the **Specification Version in Use** attribute to 0, the ONU should presume that the **Verizon OpenOMCI** specification is not supported and employ the common ITU-T G.988 set of the OMCI managed entities, attributes, and features.

Note: Even if this value is not explicitly set by the OLT, the use of OpenOMCI managed entities is not actively rejected by the ONU.

Actions

Get, set

Attribute Value Change Notifications

None

Alarms

None

4.20.12.2 Watchdog Configuration Data

ME class: 65403

This managed entity communicates the ONU's watchdog capabilities, which are applicable to the rogue ONU behavior prevention. The ONU implements a watchdog as a self-policing function to monitor its own upstream transmission parameters and to detect potentially adverse conditions to prevent behaviors that can manifest themselves as rogue from the OLT perspective. This managed entity is autonomously instantiated by the ONU upon instantiation of the ANI-G managed entity.

The following attribute values provided by this managed entity shall be implemented as shown in more detail below:

- **Managed Entity ID**
- **Upstream Transmission Timing Drift Self-monitoring Capability**
- **Upstream Transmission Wavelength Drift Self-monitoring Capability**
- **Mean In-channel Optical Power Self-monitoring Capability**
- **Mean Out-of-channel Optical Power Spectral Density Self-monitoring Capability**
- **Mean Out-of-channel Optical Power Spectral Density "when not enabled" Self-monitoring Capability**

Note: This managed entity is implemented as a template only.

Managed Entity ID

2 bytes, read-only

Through the identical managed entity ID this managed entity is implicitly associated with an instance of the **ANI-G** managed entity.

This value is configured in the default MIB initialization file.

Upstream Transmission Timing Drift Self-monitoring Capability

4 bytes, read-only

This attribute refers to the timing drift of the upstream transmission, expressed as an absolute value measured in the bit periods with respect to the nominal upstream line rate of 2.48832 Gbit/s, regardless of the actual upstream line rate of the ONU. The first two bytes of a four-byte structure contain a minimum monitored value (zero); the second two bytes carry the maximum monitored value.

The value of 0xFFFFFFFF indicates that the self-monitoring capability is not supported.

Note: Set to 0xFFFFFFFF.

Upstream Transmission Wavelength Drift Self-monitoring Capability

4 bytes, read-only

This attribute refers to the frequency drift of the upstream transmission, expressed as an absolute value measured in units of 1 GHz. The first two bytes of a four-byte structure contain a minimum monitored value (zero); the second two bytes carry the maximum monitored value.

The value of 0xFFFFFFFF indicates that the self-monitoring capability is not supported.

Note: Set to 0xFFFFFFFF.

Mean In-channel Optical Power Self-monitoring Capability

4 bytes, read-only

This attribute refers to the mean optical launch power of the burst-mode transmitter in the operating upstream wavelength channel, expressed as 2's complement in units of 0.1 dBm. The first two bytes of a four-byte structure contain the minimum monitored value. The second two bytes carry the maximum monitored value.

The value of 0xFFFFFFFF indicates that the self-monitoring capability is not supported.

Note: Set to 0xFFFFFFFF.

Mean Out-of-channel Optical Power Spectral Density Self-monitoring Capability

4 bytes, read-only

This attribute refers to the out-of-channel optical power spectral density (OOC-PSD) with respect to the current operating upstream wavelength channel and the spectral averaging interval of 15 GHz, expressed as a positive value in units of 0.1 dBm. The first two bytes of a four-byte structure contain the minimum monitored value. The second two bytes carry the maximum monitored value.

The value of 0xFFFFFFFF indicates that the self-monitoring capability is not supported.

Note: Set to 0xFFFFFFFF.

Mean Out-of-channel Optical Power Spectral Density "when not enabled" Self-monitoring Capability

4 bytes, read-only

This attribute refers to the optical power spectral density when not enabled (WNE-PSD) with respect to the spectral averaging interval of 15 GHz, expressed as a positive value in units of 0.1 dBm. The first two bytes of a four-byte structure contain the minimum monitored value. The second two bytes carry the maximum monitored value.

The value of 0xFFFFFFFF indicates that the self-monitoring capability is not supported

Note: Set to 0xFFFFFFFF.

Actions

Get

Attribute Value Change Notifications

None

Alarms

None

4.20.12.3 Watchdog PMHD

ME class: 65404

This managed entity reports the results of the ONU's self-monitoring of its transmission parameters. Each attribute is an appropriately filtered value based on the most recent observations. Instances of this managed entity are instantiated and deleted by the OLT.

The following attribute values provided by this managed entity shall be implemented as shown in more detail below:

- **Managed Entity ID**
- **Interval End Time**
- **Threshold Data 1/2 ID**
- **Upstream Transmission Timing Drift**
- **Upstream Transmission Wavelength Drift**
- **Mean In-channel Optical Power**
- **Mean out-of-channel optical power spectral density**
- **Mean optical power spectral density "when not enabled" short time scale**
- **Mean optical power spectral density "when not enabled" long time scale**

Note: This managed entity is implemented as a template only.

Managed Entity ID

2 bytes, read, set-by-create

Through the identical ME ID this managed entity is implicitly associated with an instance of the **ANI-G** managed entity.

Interval End Time

1 byte, read-only

This attribute identifies the most recently finished 15-minute interval.
With respect to the Watchdog PMHD managed entity, this attribute is not used.

Note: Set to 0.

Threshold Data 1/2 ID

2 bytes, read, write, set-by-create

This attribute points to an instance of the threshold data 1 and 2 managed entities that contains PM threshold values.

Upstream Transmission Timing Drift

2 bytes, read-only

This attribute reports the self-monitored value based on the recent observations and expressed as an absolute value measured in units of 2.48832 Gbit/s bit periods.

Upstream Transmission Wavelength Drift

2 bytes, read-only

This attribute reports the self-monitored value based on the recent observations and expressed as an absolute value measured in units of 1 GHz.

Mean In-channel Optical Power

2 bytes, read-only

This attribute reports the self-monitored value based on the recent observations and expressed in units of 0.1 dBm.

Mean out-of-channel optical power spectral density

2 bytes, read-only

This attribute reports the self-monitored value (OOC-PSD) based on the recent observations and expressed as a positive value in units of 0.1 dBm with respect to the spectral averaging interval of 15 GHz.

Mean optical power spectral density “when not enabled” short time scale

2 bytes, read-only

This attribute reports the self-monitored short-time peak value (WNE-PSD) based on the recent observations and expressed as a positive value in units of 0.1 dBm with respect to the spectral averaging interval of 15 GHz.

Mean optical power spectral density “when not enabled” long time scale

2 bytes, read-only

This attribute reports the self-monitored long-time average value (WNE-PSD) based on the recent observations and expressed as a positive value in units of 0.1 dBm with respect to the spectral averaging interval of 15 GHz.

Actions

Create, delete, get, set, get current data

Attribute Value Change Notifications

None

Threshold Crossing Alerts

The following thresholds are checked and an alert is sent if a threshold is exceeded.

Table 71 Watchdog PMHD TCA List

Alarm Number	Threshold Crossing Alert	Threshold Values Attribute Number
0	Timing drift warning	1
1	Timing drift shut-off	2
2	Wavelength drift warning	3
3	Wavelength drift shut-off	4
4	–	–
5	Transmit optical power too high warning	6
6	Transmit optical power too high shut-off	7
7	OOC-PSD too high warning	8
8	OOC-PSD too high shut-off	9
9	Short time scale WNE-PSD too high warning	10
10	Short time scale WNE-PSD too high shut-off	11
11	Long time average WNE-PSD too high warning	12
12	Long time average WNE-PSD too high shut-off	13

4.20.12.4 Flexible Configuration Status Portal

ME class: 65420

This managed entity provides a means to configure and report status for a wide range of services and applications running on the ONU. The managed entity is designed to support IP-based (streaming message) and OMCI-based (block-oriented) management protocols. Only one method would be used for a given service type.

This managed entity does not have any service specific attributes, deferring these to the details of the configuration table or to IP datagrams. An instance of this managed entity is created by an ONU for each entity that can be supported by this managed entity. The OLT can create additional instances as needed.

The configuration portal's structure is not visible to OMCI. However, the configuration portal is modeled as a table, each row holding 25 bytes. The first byte of the row is the row index and is not part of the configuration information.

Note: This managed entity is implemented as a template only.

Managed Entity ID

2 bytes, read, write, set-by-create

This value is configured in the default MIB initialization file for instances that are autonomously created or provided by the OLT, if the OLT creates an instance of this managed entity.

Service Instance

4 bytes, read, write, set-by-create

This attribute identifies the specific service type supported by the given instance this managed entity. The Service Instance consists of three parts.

The first two bytes contain the Service Type ID, as documented below.

The third byte identifies the transport protocol on top of the IP protocol to be used to carry the service.

- 0 = UDP
- 1 = TCP
- 2 = SCTP
- 0xFF = not used (for message-based configuration)

The fourth byte identifies the instance of the Service Type ID on the associated managed entity class and managed entity instances. Typically only one instances of a service will be instantiated on a given entity.

The mapping of the Service Type ID to a service follows IANA "Service Name and Transport Protocol Number Registry". Services that are not in the registry are assigned specific code points. This specification supports the following Service Type IDs.

- NETCONF over SSH, 830, TCP
- IPFIX, 4739, UDP
- MCIT, 1028, 0xFF
- XFP/SFP MSA Data, 0, 0xFF

Configuration Method

2 bytes, read, write, set-by-create

This attribute indicates the configuration/status management method used for this service instance. The two management methods are IP or message based. The management methods for the configuration and the status can be independently set, although typically both would be set to the same value. Bit value of 0 means the configuration and status is carried over an IP path. The value of 1 means the configuration and status is carried over the configuration and status portals.

Network Address

2 bytes, read, write, set-by-create

This is a pointer to a Network Address managed entity. Used if Bit 1 and/or Bit 2 of the Configuration Method is set to 0. Null means that the network address is not used.

Administrative State

1 byte, read, write, set-by-create

This attribute allows to administratively control the service over OMCI. The value of 0x00 indicates that the service is unlocked, that is, allowed to operate normally. The value of 0x01 indicates the service is locked, that is, should stop. As a rule, the default value upon instantiation is 0x01.

Operational State

1 byte, read

This value represents the operational state of the service. The value of 0 indicates normal operation, 1 indicates stopped. Other values are Service Type dependent.

Cause of Last Abnormal Halt

2 bytes, read, write, set-by-create

This attribute represents a service type specific code for the last abnormal halt of the service. It complements the operational state attribute to reflect the cause for the most recent abnormal halt of the service. Writing this attribute clears the setting. The default value is 0xFFFF (halt has not occurred yet).

Configuration Portal Update Available

2 bytes, read, write, set-by-create

This attribute indicates to the service that new data is available in the Configuration Portal. The default value is 0. The OLT sets the attribute to 1 to indicate new data in the Configuration Portal. This value should be set to non-zero only after the Configuration Portal Result attribute is zero. The OLT sets the attribute to 0 after it 1 reads the Configuration Portal Result attribute having a non-2 zero value.

Configuration Portal Table

25 x N bytes, read, write, set-by-create

This attribute contains configuration information directed from the OLT towards the service. This attribute is used if Bit 1 of Configuration Method is set to 1. Each row of the table is defined as

- Index into the table (1 byte)
- Configuration Information (24 bytes)

The format of the message is service-type dependent.

Configuration Portal Result

1 byte, read, write, set-by-create

This attribute reports the status of the service reading the configuration information via the Configuration Portal. The code points are:

- 0: Not read
- 1: Read
- 2: Internal error on read

The ONT sets the value to 1 or 2 after the Configuration Portal Update Available is set to a non-zero value and after it has read the data in the Configuration Portal. The ONT sets the value to 0 after it reads the Configuration Portal Update Available attribute is zero.

Status Message Available

2 bytes, read, write, set-by-create

This attribute indicates that new data is available in the Status Message. The default value is 0. The ONU sets the attribute to N to indicate that the first N bytes of the Status Message contain new data and should occur only when the Status Message Result attribute is zero. The ONU sets the attribute to 0 after the Status Message Result attribute has been read and shows a non-zero value.

Status Message

x bytes, read

This attribute contains status information about the service. The format of the message is a TLV based set of content that is service-type dependent. The expected format of the contents is zero or more TLVs of the form:

- Byte offset 0 to 1: Tag class identifier
- Byte offset 2 to 3: Tag item identifier
- Byte offset 4 to 5: Contents length
- Byte offset 6 to N: Information contents

The get/get next sequence must be used with this attribute since its size is unspecified. Upon managed entity instantiation, the attribute is set to 0 (one byte).

Status Message Result

1 byte, read, write, set-by-create

This attribute reports the status of the OLT reading the Status Message. The code points are:

- 0: Not read
- 1: Read

The OLT sets the value to 1 after the Status Message Available is set to 1 and after it has read the data in the Status Message. The OLT sets the value to 0 after it reads the Status Message Available attribute to be zero.

Associated ME Class

2 bytes, read, write, set-by-create

This attribute identifies the managed entity class that this managed entity is associated with.

Associated ME Class Instance

2 bytes, read, write, set-by-create

This attribute identifies the managed entity ID of the of the associated managed entity.

Actions

Create, Delete, Get, Set, Get Next, Set table (optional)

Attribute Value Change Notifications

5, 9, 10

Alarms

0, 1, 2, 3

4.20.12.5 Flexible Configuration Status Portal PMHD

This managed entity provides statistics for the Flexible Configuration Status Portal. This managed entity is created by the OLT and implicitly associated with the instance of the Flexible Configuration Status Portal.

This managed entity is not implemented.

4.20.12.6 MAC Swap Loop Configuration

This managed entity is not implemented.

4.20.12.7 MAC Swap Loop Monitor

This managed entity is not implemented.

4.20.12.8 Extended Remote Debug

This managed entity is not implemented.

5 PLOAM Protocol

The Physical Layer Operation, Administration, and Maintenance (PLOAM) management protocol of the PON interface is used to initiate a connection between the ONU and the OLT. This protocol is not handled by the software, but by the and firmware included in the PON IP hardware module.

There is no need for the software to be aware of this protocol, but it is possible to forward PLOAM to the software for debugging purposes or to extend the PLOAM protocol by vendor-specific functionality.

Use the PON library function `fapi_pon_ploamd_cfg_set` to select specific PLOAM downstream messages to be forwarded to the software for vendor-specific handling.

See **Chapter 6.4** for more detail.

5.1 PLOAM Message Overview

PLOAM messages are specific for different PON technologies. **Table 72** and **Table 73** show the messages that are supported for the TDM and TWDM PON technologies.

Table 72 PLOAM Downstream Message Overview

PLOAM Message ID	G-PON (G.984)	XG-PON (G.987)	XGS-PON (G.9807)	NG-PON2 (G.989)
1	Upstream Overhead	Profile	Burst Profile	Burst Profile
2	Assign ONU ID	Assign ONU ID	–	–
3	Ranging Time	Ranging Time	Assign ONU ID	Assign ONU ID
4	Deactivate ONU ID	Deactivate ONU ID	Ranging Time	Ranging Time
5	Disable Serial Number	Disable Serial Number	Deactivate ONU ID	Deactivate ONU ID
6	Encrypted Port ID	–	Disable Serial Number	Disable Serial Number
7	Request Password	–	–	–
8	Assign AllocID	–	–	–
9	No Message	Request Registration	Request Registration	Request Registration
10	Popup	Assign AllocID	Assign AllocID	Assign AllocID
11	Request Key	–	–	–
12	Configure Port ID	–	–	–
13	Physical Equipment Error	Key Control	Key Control	Key Control
14	Change Power Level	–	–	–
15	PST	–	–	–
16	BER Interval	–	–	–
17	Key Switching Time	–	–	–
18	Extended Burst Length	Sleep Allow	Sleep Allow	Sleep Allow
19	–	–	–	Calibration Request
20	–	–	–	Adjust Tx Wavelength
21	–	–	–	Tuning Control
22	–	–	–	
23	–	–	–	System Profile
24	–	–	–	Channel Profile

Table 72 PLOAM Downstream Message Overview (cont'd)

PLOAM Message ID	G-PON (G.984)	XG-PON (G.987)	XGS-PON (G.9807)	NG-PON2 (G.989)
25	–	–	–	Protection Control
26	–	–	–	Change Power Level
27	–	–	–	Power Consumption Inquire
28	–	–	–	Rate Control (P2P only)
29	–	–	Reboot ONU	Reboot ONU

Table 73 PLOAM Upstream Message Overview

PLOAM Message ID	G-PON (G.984)	XG-PON (G.987)	XGS-PON (G.9807)	NG-PON2 (G.989)
1	Serial Number ONU	Serial Number ONU	Serial Number ONU	Serial Number ONU
2	Password	Registration	Registration	Registration
3	Dying Gasp	–	–	–
4	No Message	–	–	–
5	Encryption Key	Key Report	Key Report	Key Report
6	Physical Equipment Error	–	–	–
7	PST	–	–	–
8	Remote Error Indication	–	–	–
9	Acknowledge	Acknowledge	Acknowledge	Acknowledge
…				
16	–	Sleep Request	Sleep Request	Sleep Request
…				
26	–	–	–	Tuning Response
27	–	–	–	Power Consumption Report
28	–	–	–	Rate Response

6 PON Library

The PON library (also referred to as PON functional application programming interface, PON library, `pon_lib`) provides access to the device PON-specific functions. These are function mainly related to the PON IP hardware module and its integrated processor which runs the PON firmware. The PON library version is related to the PON firmware version in terms of functionality and internal interfacing. Make sure the matching versions are used as delivered with the 10G PON Chipset System Package and do not mix versions derived from different software deliveries.

The PON library uses a mailbox interface to access the PON IP module. These messages must exclusively be handled by the PON mailbox driver (`pon_mbox_drv`) and never accessed by vendor-specific code directly.

6.1 PON Library Functionality

The basic functionality of the PON library is described in this section. A get call is used as an example to show the internal flow and dependencies.

A simple get library call works this way:

1. The function `fapi_pon_*_get(struct <type> param)` gets called by external code.
2. This function calls the `fapi_pon_generic_get()` function with the related command ID.
   In addition, there is also a callback handler provided and the `param` parameter.
3. The function `fapi_pon_generic_get()` calls some other methods to create a read message for the firmware. This message call does not include any payload as it is intended to read the specified data back from a response message.
4. The request data is sent to the `pon_mbox_drv` driver through a generic Netlink and the request is forwarded to the firmware.
5. A callback handler is registered and must be called to evaluate the answer.
6. The `nl_recvmsgs()` function is called and handles the reception of Netlink messages.
7. When the firmware answers to the message, the `pon_mbox_drv` driver gets the message and creates a new Netlink message as a response to the request.
   a) The `pon_mbox_drv` driver matches the request to the response by checking the response sequence number. This must match the request sequence number.
   b) The answer is only sent to the process which has issued the request.
8. The callback handler (`cb_valid_handler()`) registered in step 5 is called by the Netlink library with the received data.
9. The data is extracted and given to the callback handler registered in step 2.
   It gets the payload received through the firmware message and the `param` value (structure) given to `fapi_pon_generic_get()` in step 2.
10. The callback handler registered in step 2 converts the contents received from the firmware into the format used by the PON library.
    a) It is checked whether the size matches the expected size of the message.
    b) The result is written to the `param` structure provided by the external caller.
11. Both callback functions terminate.
    The Netlink message itself is freed.
12. The function `fapi_pon_generic_get()` returns with a status code.
13. The function `fapi_pon_*_get(struct <type> param)` returns with a status code.
    The `param` structure has been updated in the callback function.

6.2 PON Library Command Line Access

All functions of the PON library are available as CLI debug functions to ease the software development.

The CLI functions are by default not compiled into productive images for code security reasons and the OEM software is strictly advised to follow this default. For debug and development binaries, the CLI functions can be compiled in. See **Section 14.3.3 PON Command Line Functions** for details.

6.3 PON Alarm Handling

The PON adapter receives alarms indications from the PON mailbox driver (pon_mbox_drv) through the PON library (`pon_lib`). There are two types of alarm provided:

- Static or level alarms
- Dynamic or edge alarms

Each alarm is identified by a unique identifier (`alarm_id`). A list of all supported alarm types and their respective IDs is available in **Section 6.3.1**.

Level Alarms

These alarms have a distinct but variable duration time. For example, a loss of signal (LOS) starts when the light disappears at the optical receiver and ends when the light is back. At both times an event is generated, one for alarm set and another for alarm cleared. The status of all level alarms is read though a PON library function (`pon_alarm_status_get(alarm_id)`).

Edge Alarms

These alarms do not have a duration and only happen at a time. For example, to signal a change of the OMCI integrity key. These alarms are only indicated when they happen and there is no status to be read.

6.3.1 Alarm Types

Table 74 lists the supported alarm types. The numbers not listed are reserved for future use or debug purposes.

Table 74 PON Alarm Types

Alarm ID	Short Name	Description	Mode
0000_H	**LOS**	Loss of signal. This is caused by a missing or incorrect optical signal, or when the SerDes loses synchronization.	Level
0001_H	**LOF**	Loss of frame (G984.3 only)	Level
0002_H	**LODS**	Loss of downstream synchronization	Level
0003_H	**SF**	Signal fail	Level
0004_H	**SD**	Signal degrade	Level
0005_H	**LCDG**	Loss of GEM channel delineation *Note: Although this is defined only in ITU-T G.984.3, the alarm is provided in all ITU PON operational modes. The alarm is cleared not earlier than 1 ms after setting to avoid alarm bursts.*	Level
0006_H	**TF**	Transmitter failure	Level
0007_H	**SUF**	Start-up failure	Level
0008_H	**MIS**	Link mismatch (G984.3 only)	Level
0009_H	**O7**	Emergency stop state entered (PLOAM state O7/O7.1/O7.2)	Level
$000B_H$	**CLK**	Clock alarm reported by the SoC	Level

Table 74 PON Alarm Types (cont'd)

Alarm ID	Short Name	Description	Mode
$000C_H$	**FSR**	Free spectral range mismatch (TWDM only)	Level
$000D_H$	**CSP**	Channel spacing mismatch (TWDM only)	Level
$000E_H$	**MSE**	Maximum spectral excursion mismatch (TWDM only)	Level
$000F_H$	**UPM**	Unknown PLOAM message	Edge
0010_H	**DACT**	Deactivate ONU message received	Edge
0011_H	**PEE**	Physical equipment error (G984.3 only)	Level
0012_H	**RDI**	Remote defect indication (G984.3 only)	Edge
0013_H	**ROG**	Rogue ONU detected by the automatic self-detection	Edge
0015_H	**FWA**	Firmware alarm (for example if the internal mailbox is full on the firmware side)	Edge
0016_H	**OIKC**	OMCI integrity key has changed	Edge
0017_H	**PCE**	Persistent crossbar access error. This is not expected to happen on a healthy system. When it happens, reboot.	Level
0018_H	**PSE**	Persistent PON SerDes error. This is not expected to happen on a healthy system. When it happens, reboot.	Level
$001A_H$	**ASC**	The mutual authentication status has changed.	Edge
$001E_H$	**EQDEL**	ONU response time (equalization delay and ONU response time) value change	Edge
$001F_H$	**CPI_TO**	Channel Profile Index time-out event (TWDM only)	Edge
0100_H	**FWI**	Forced wake-up indication received from the OLT while in a sleep mode	Edge
0101_H	**FWVERR**	Firmware version error. The loaded firmware does not fit to the PON IP hardware version.	Edge
0102_H	**NG_ATWLPI**	NG-PON2 Message AdjustTXWavelength parameter inconsistency (TWDM only)	Edge
0108_H	**RTERR**	Ranging time error - the OLT has sent a ranging time value which cannot be handled by the firmware.	Edge

6.3.2 PON Library Alarm Handling

The PON firmware issues a number of event notifications (alarms) registered by the software to react on them. For example, the LOS alarm is used to control a LED for line status indication.

The implementation of the alarm handling in the user software is described here, using the PON debug daemon (`pond`) code. The source code is located at `/pon_lib/daemon/pond.c`.

These parts are essential for the alarm handling.

1. Include the library files

```
#include "fapi_pon_os.h"
#include "fapi_pon.h"
#include "fapi_pon_error.h"
#include "fapi_pon_alarms.h"
```

2. Open the PON context.
```
ret = fapi_pon_open(&cfg.fapi_ctx);
if (ret != PON_STATUS_OK) {
        fprintf(stderr, "creating pon context failed\n");
        return EXIT_FAILURE;
}
```
3. Connect the event listener.
```
ret = fapi_pon_listener_connect(cfg.fapi_ctx, &cfg);
if (ret != PON_STATUS_OK) {
        fprintf(stderr, "connecting event listener failed\n");
        return EXIT_FAILURE;
}
```
4. Register the callback function. Here, `print_active_alarms`.
```
fapi_pon_register_alarm_report(cfg.fapi_ctx, pond_print_active_alarms);
```
5. Enter an endless loop to wait for events.
```
while (listen) {
        ret = fapi_pon_listener_run(cfg.fapi_ctx);
        if (ret != PON_STATUS_OK)
                break;
}
```

The callback function must be provided by the user application and perform the actions required for the specific event. In this example, there is a printout for the currently active alarms. Typical events handled in PON applications are:

- Loss of signal (LOS) or loss of downstream synchronization (LODS)
- Change of the OMCI integrity key (OMCI_IK)
- Change of the PLOAM state (for example, leaving or entering the operational state O5, entering or leaving the emergency stop state O7)
- Reception of unknown (unhandled) PLOAM downstream messages

Alarm Disabling

Table 74 lists the alarms that the `fapi_pon_alarm_cfg_set` function is able to individually disable. The same function is used to enable the alarms.

Handling of Multiple Callbacks

Only one single callback handler is registered per function at a time. For example, a call to `fapi_pon_register_alarm_report()` removes the function which was previously registered to that specific callback. However, in this example, this does not remove a function registered to `fapi_pon_register_alarm_clear()`.

When more than one callback handler is required for a specific event, multiple context instances must be created.

Dying Gasp Alarm Suppression

In contrast to other alarms, a Dying Gasp alarm does not only trigger a local alarm to the PON firmware but also sends this information to the OLT. This is done either by sending a dedicated PLOAM message (for G.984 operation mode) or by setting a flag in the upstream XGTC header (for other ITU PON operation modes).

When the power supply voltage reference is not available in the application, such as in an SFP, this alarm must be suppressed to avoid false indication to the OLT.

This is configured in the UCI file `/etc/config/optic` either:

- by running the UCI command:
  ```
  uci set optic.common.dg_dis=1
  uci commit
  ```
- or by directly editing the file.

6.3.3 Automatic Rogue ONU Detection

There are two options to detect a rogue ONU state automatically. This avoids service interruption on the ODN when the ONU is defective and sends and optical signal outside of assigned upstream time slots.

6.3.3.1 External Rogue ONU Detection

The modern PMD devices are able to detect possible rogue ONU state at the optical edge of the ONU and signal this through the `TX_FALUT` signal to the PON TC layer. This detection capability is used to react accordingly and shut down the transmission signal immediately.

Refer to the PMD vendor data sheet for additional details regarding the detection.

This mode is recommended and is selected in `/etc/config/optic` by selecting `rogue_auto_en '2'`.

See **Section 14.7.1** for testing hints.

6.3.3.2 Rogue ONU PLOAM State Handling

In addition, when a rogue ONU state has been detected, the firmware-controlled PLOAM state machine is forced to move into the emergency stop state (O7), which in turn triggers the related software event (O7).

This is enabled by setting bit 4 of the PON interoperability mask in `/etc/config/gpon` (`iop_mask`). See **Section 2.4.3.3**, **Table 2** for details.

This ensures that the transmitter is not re-enabled after a reboot or power cycle, when the rogue ONU state has been detected. It is up to the operator requirements, under which conditions the emergency stop state must be cleared. This is under control of the OLT. **Figure 125** shows an example of the state machine enhancement for the XGS-PON operation mode.

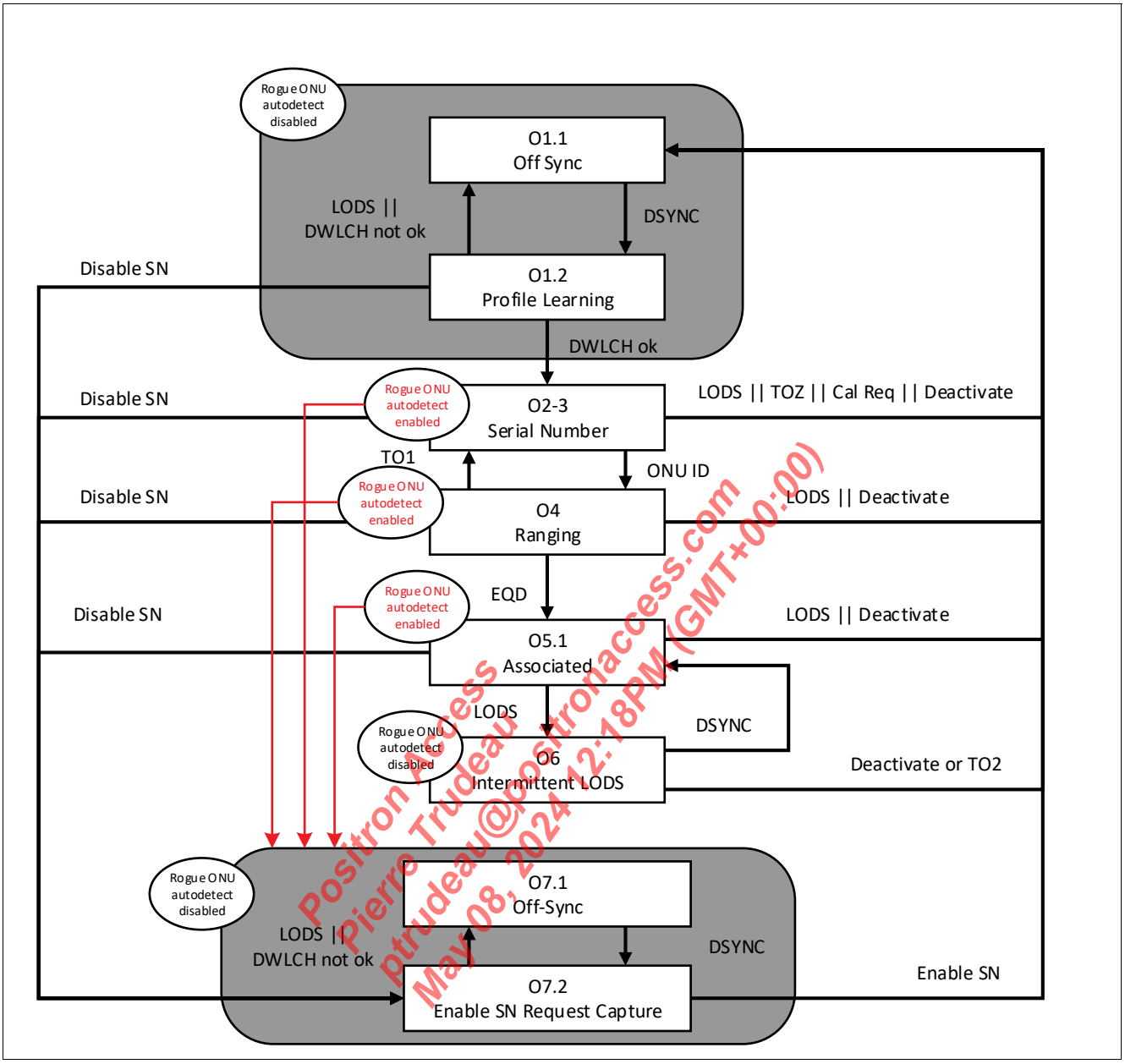


Figure 125 Rogue ONU Detection in PLOAM State Machine

6.4 Vendor-specific PLOAM Message Handling

When the OLT sends vendor-specific (non-standard) PLOAM downstream messages (PLOAMd) or expects to receive such PLOAM upstream messages (PLOAMu), this cannot be handled by the PON firmware which controls the regular PLOAM message protocol. The firmware is unaware of these message types and thus this must be covered by vendor-specific software.

6.4.1 Vendor-specific PLOAM Downstream Messages

Up to 16 selectable PLOAMd message types are defined to be handled by the software. The function `fapi_pon_ploamd_cfg_set` is used to configure the related PLOAM message type IDs.

For each registered vendor-specific PLOAMd message, the PON firmware acknowledges such messages as OK (Completion_code = 0x00). Unregistered unknown PLOAMd messages are acknowledged by sending unknown message type (Completion_code = 0x03).

When a configured PLOAMd message is received, an event is triggered which must be registered by the dedicated software handler. See **Section 8.2** for details on event handling.

This function is also used to monitor dedicated PLOAMd messages for debugging purposes.

To monitor all PLOAMd and PLOAMu messages, see the `fapi_pon_ploam_log_enable` and `fapi_pon_ploam_log_disable` functions.

6.4.2 Vendor-specific PLOAM Upstream Messages

The PLOAMu messages cannot be sent to the OLT. These messages are under exclusive firmware control to avoid interference with the PLOAM activation state machine.

6.5 PLOAM State Coding

The states of the PLOAM activation state machine depend on the active PON operation mode. **Table 75** shows the two-digit decimal numerical coding used to access the PLOAM state machine status.

Table 76 shows the enhanced set of states that are applicable to NG-PON2 operation mode.

Table 75 PLOAM State Coding – Single Wavelength Pair

PLOAM State Acronym	Code Point	Description	Used by		
			G.984	G.987	G.9807
O0	00_D	Power-up state	x		
O1	10_D	Initial state	x	x	
O1.1	11_D	Off-sync state			x
O1.2	12_D	Profile learning state			x
O2	20_D	Stand-by state	x		
O2.3	23_D	Serial number state		x	x
O3	30_D		x		
O4	40_D	Ranging state	x	x	x
O5	50_D	Operation state	x	x	
O5.1	51_D	Associated state			x
O6	60_D	Popup state	x		
		Intermittent LOS state		x	x
O7	70_D	Emergency stop state	x	x	x

Table 76 PLOAM State Coding – Multiple Wavelength Pairs

PLOAM State Acronym	Code Point	Description	Used by
			G.989
O1.1	11_D	Off-sync state	x
O1.2	12_D	Profile learning state	x
O2.3	23_D	Serial number state	x
O4	40_D	Ranging state	x
O5.1	51_D	Associated state	x
O5.2	52_D	Pending state	x
O6	60_D	Intermittent LOS state	x
O7	70_D	Emergency stop state	x
O7.1	71_D	Emergency stop off-sync state	x
O7.2	72_D	Emergency stop in-sync state	x
O8	80_D	Downstream tuning state	x
O8.1	81_D	Downstream tuning off-sync state	x
O8.2	82_D	Downstream tuning profile learning state	x
O9	90_D	Upstream tuning state	x

6.6 PON Wavelength Switching

This chapter applies to NG-PON2 operation only.

The wavelength switching is done by accessing the PMD device over the I²C bus interface. Each time the PON firmware needs to select a new downstream or upstream wavelength, a firmware message is sent to the software.

The message `TWDM_DS_WL_CONFIG` selects a new downstream wavelength while the message `TWDM_US_WL_CONFIG` selects a new upstream wavelength.

The software handles these messages as events and performs the required I²C configuration commands. The related code can be found in `pon_lib/adapter/fapi_pon_pa_event.c`.

6.6.1 Supported Wavelengths

The supported wavelength index values are defined in the UCI configuration `optic.twdm.channel_mask`.

For example, a typical value of 0x0F indicates that the wavelength index values 0, 1, 2, and 3 are available for selection. It is assumed that the optical transceivers support optical wavelengths in a symmetrical manner (the same index values of transmit and receive wavelength are supported).

6.6.2 Wavelength Switching Methods

Due to the fact that the wavelength switching commands are not uniquely defined, different methods must be implemented to serve different PMD types.

The reference implementation provides different methods (`fapi_pon_twdm_write_method<n>`).

The applicable method is selected in `handle_twdm_wl_config` based on the UCI configuration `optic.twdm.config_method`.

If a PMD is used that needs a configuration method that is not yet supported, the code must be enhanced accordingly.

6.6.2.1 Wavelength Switching Method 0

This is a dummy method that just accepts the switching request from and acknowledges it to the PON firmware without performing any wavelength change. This can be used for transceiver-independent software development.

This method is used if no valid switching method has been configured for the transceiver that has been detected.

6.6.2.2 Wavelength Switching Method 1

This method is used for this transceiver type:

- Hisense LTW2601 (ligentphotonics-ltw2601cbc)

6.6.2.3 Wavelength Switching Method 2

This method is used for this transceiver type:

- Lightron LWEKR-Rxx8-A (lightroninc-0013c5-lwekrrxx8a)

6.6.2.4 Wavelength Switching Method 3

This method is used for this transceiver type:

- PICadvanced

7 PON Image Library

The PON image library (`pon_img_lib`) provides the functions that are needed to perform a software image upgrade. To ensure a safe software upgrade process, two software images are present in the flash memory. The image upgrade process is aligned with the OMCI-based mechanism and handles the software image status variables that are stored in the U-Boot environment. Each of the two images that are stored has four properties:

- Active/inactive
- Valid/invalid
- Committed/not committed
- Version identifier

7.1 PON Library Functionality

The basic functionality of the PON library is described in this section.

The following functions are provided by the library:

- `pon_img_upgrade` - Execute the image upgrade process.
- `pon_img_active_set` - Activate or de-activate one of the images.
- `pon_img_active_get` - Check if an image is the active one.
- `pon_img_commit_set` - Commit or de-commit one of the images.
- `pon_img_commit_get` - Check if an image is committed.
- `pon_img_version_set` - Set the version string for an image.
- `pon_img_version_get` - Read the version string of an image.
- `pon_img_valid_set` - Set an image to be valid or invalid.

7.2 PON Image U-Boot Environment Variables

To store the software image properties permanently across a reboot or power cycle, these properties are stored in U-Boot environment variables. These variables are used:

- `active_bank` = A|B
- `img_activate` = A|B
- `commit_bank` = A|B
- `img_versionA` = 14-character ASCII string (used to be reported by OMCI)
- `img_versionB` = 14-character ASCII string (used to be reported by OMCI)
- `img_validA` = true|false
- `img_validB` = true|false

See **Table 43** for the OMCI value mapping.

7.3 Flash Memory Configuration

The image is stored in an external flash memory device. This memory device is connected through a serial SPI interface. It is possible to run the SPI interface in either single-lane or quad-lane operation mode.

If the selected memory device supports quad-lane operation, this must be enabled in the build configuration for U-Boot *and* for Linux.

Note: Not all types of flash memory devices are able to support the quad-lane SPI mode for initial boot ROM access. In this scenario the boot strapping must enable single lane mode for U-Boot and then use quad-lane operation mode for Linux.

8 PON Adapter

This chapter describes the handling of the PON adapter software.

8.1 PON Adapter Introduction

The PON adapter software layer is used to connect the management stack to lower layer functionality. This is intended to make the software layers independent and allow to exchange either the management stack or the underlying hardware and lower layers.

The main functionality provided by the PON adapter is:

- Configuring the hardware and software functions
- Getting information from the hardware modules
- Receiving event notifications (alarms, etc.)

The interface is generic to allow any management stack to be attached. When implementing OMCI, for example, the structure definitions follow the managed entity model as defined by OMCI to enable a one-to-one mapping of attributes.

8.2 PON Adapter Software Structure – OMCI Implementation

The OMCI software stack PON adapter implementation is based on the adapter design pattern. **Figure 126** shows the PON adapter top level interface and class view. The PON adapter interfaces are used to call the different lower layer operations for the different modules. All modules are registered using the PON adapter interfaces during start-up.

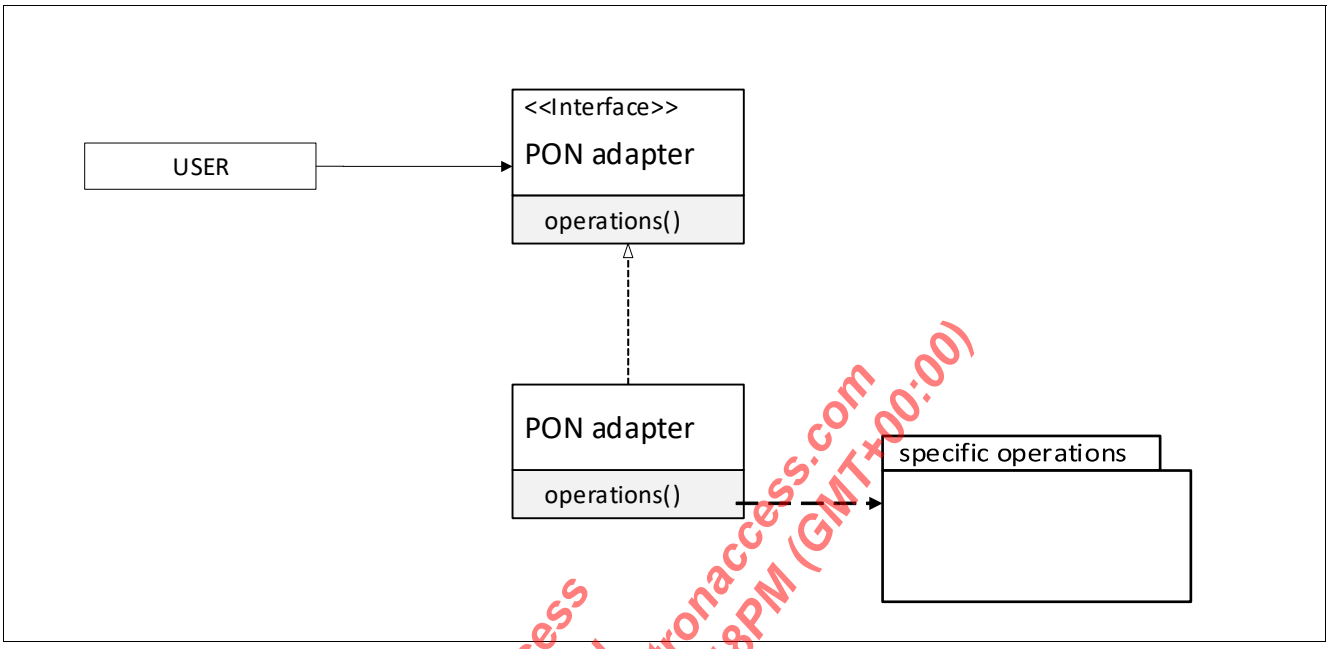


Figure 126 PON Adapter Top Level Interface

Figure 127 shows the PON adapter lower layer module registration flow of the OMCI daemon. The modules are specified in a UCI configuration file and each module is a share object and loaded with the `dlopen` function. During the OMCI daemon initialization, the init and start operation of each lower level module system specific operations are called.

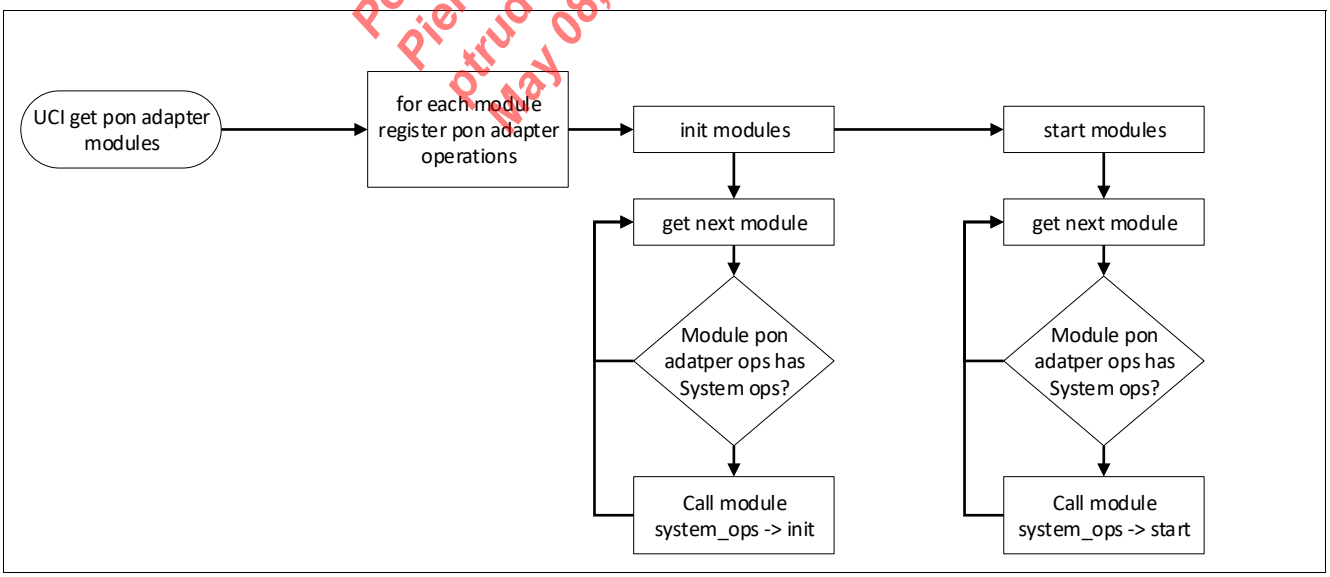


Figure 127 PON Adapter Low Level Modules Registration

The call of the init and start operations is not done immediately one of another. The OMCI daemon does other initialization in between. **Figure 128** shows an example of PON adapter lower layer operations structures. This is only a subset of the possible operations which also depend on the PON adapter lower layer module. As mentioned before, each module has and implements system specific operations used during its initialization. These operations are part of the `struct pa_ops`. All PON adapter modules provide their own specific operations base on this interface.

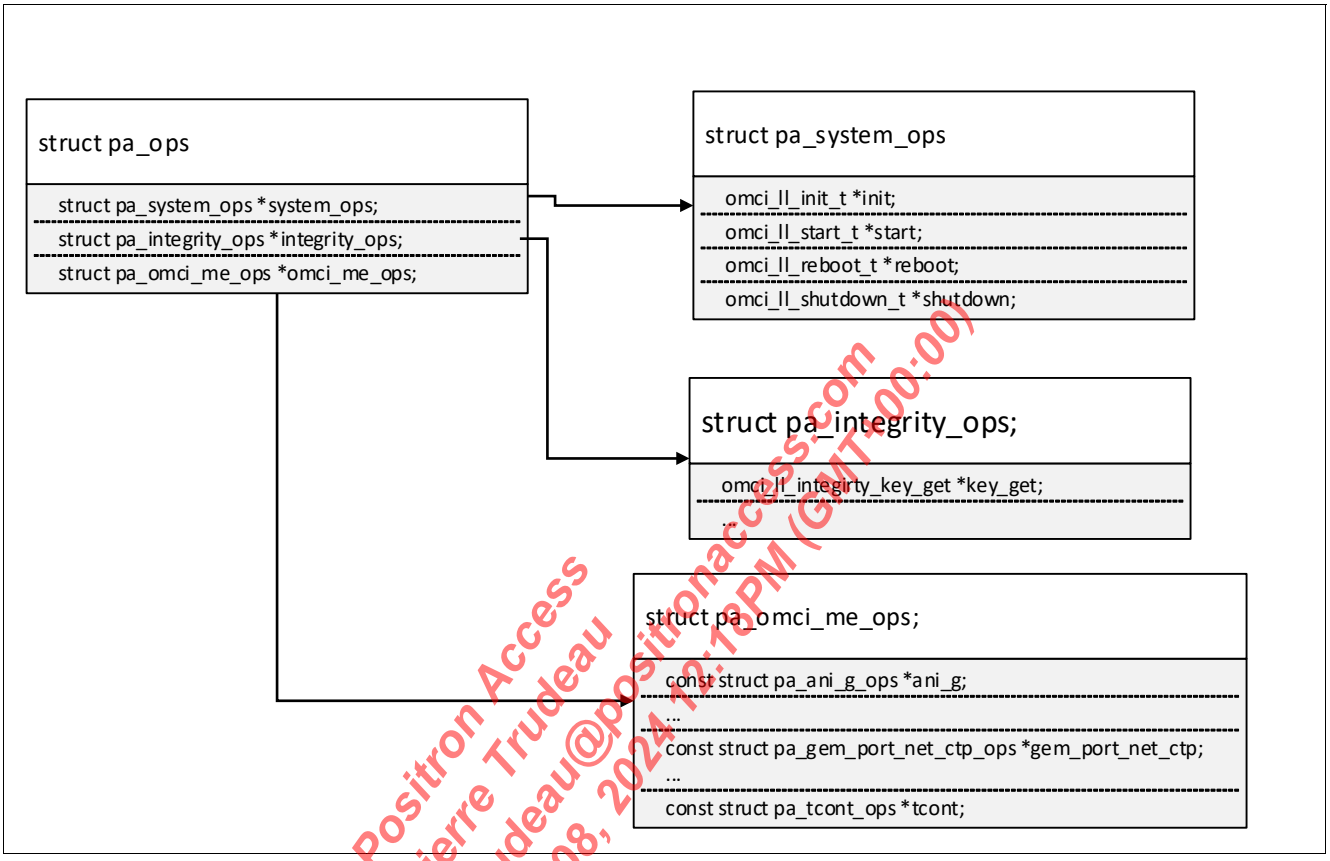


Figure 128 PON Adapter Lower Layer Operation Structures

Figure 129 shows an example of how a specific PON adapter operation of lower layer is called. The upper layer, in this case the OMCI ME T-CONT class, searches in each registered module for OMCI ME operations. When such are found, the operations structure is checked whether it contains the PON adapter T-CONT operations. At the end, this pointer to the T-CONT operations is returned to the higher layer and its specific functions are called from the ME class. On the figure, this is the `create` operation. Each ME operation structure has specific operation callbacks defined and these callbacks differ between the different MEs. One ME class can also use more than one lower level module, not only the `ponlib` as shown on the figure. This means that two modules can export different set of specific operations to higher layers implementing different functionalities. For example, one ME uses the same operation structure from both `ponlib` and `pon-net-lib` but the specific operation of each of the modules configures different things: one does PON specific configuration and the other network specific one using Linux APIs.

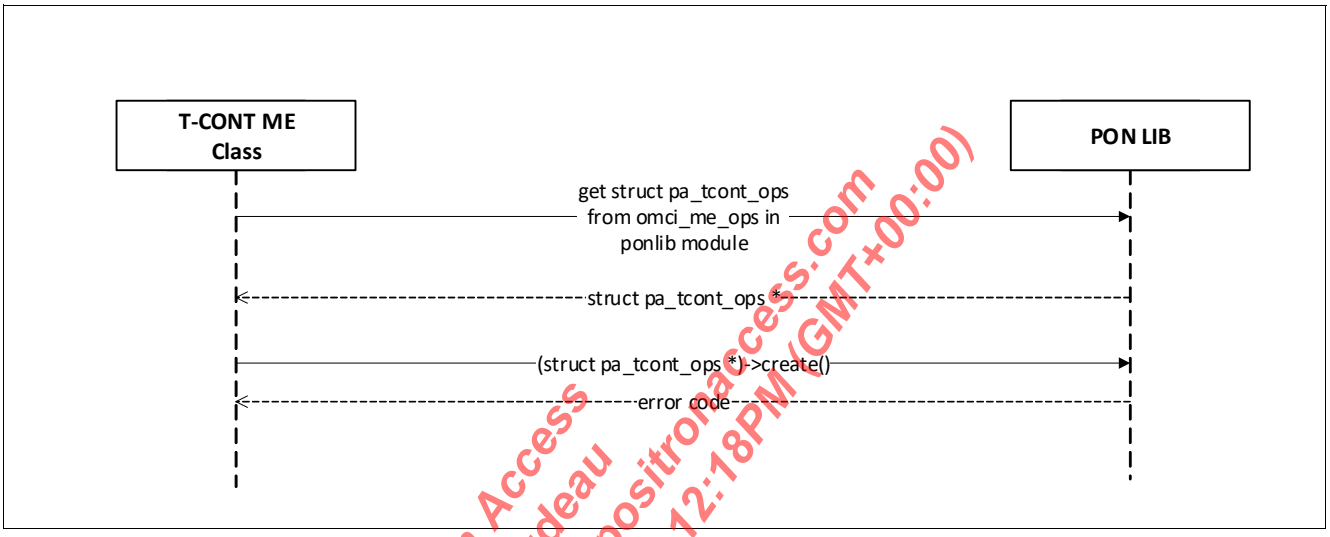


Figure 129 PON Adapter Lower Layer Operation Call Sequence

The PON adapter modules also forward events to higher layers by using the event operation structures. For example, the `ponlib` module uses the `struct pa_eh_ops` operation structure from the PON adapter interface. **Figure 130** shows the usual sequence flow. The higher layer must register its event handlers using this interface to get the events from the lower layer. It must fill the `pa_eh_ops` with the necessary callbacks.

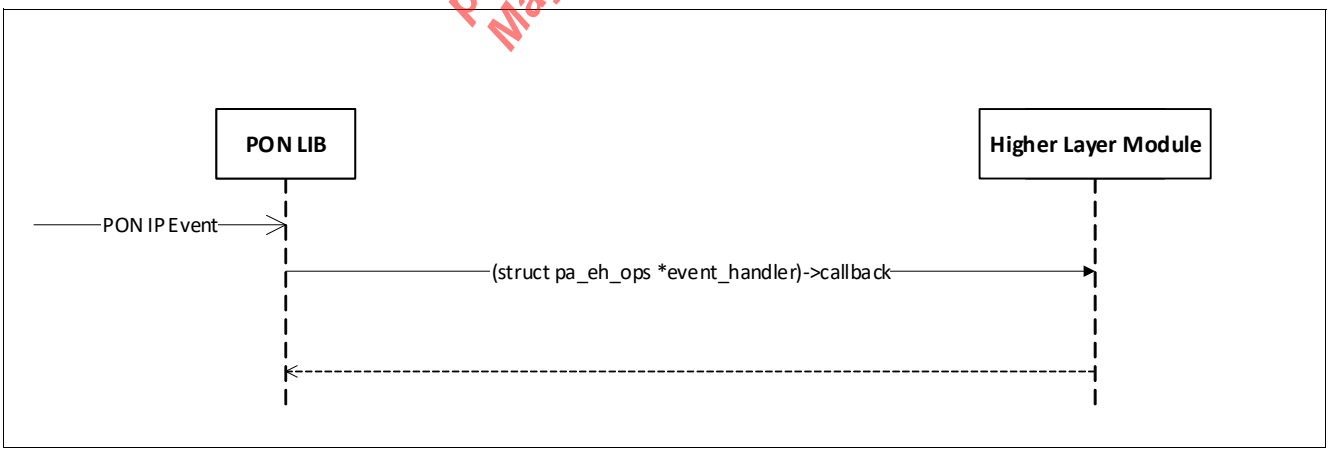


Figure 130 Handling Events in PON Adapter

Figure 131 shows the software architecture of the PON adapter layer.The PON adapter must be used when a higher layer such as the OMCI stack must access a lower layer hardware specific functionality, for example the PON mailbox driver or configuring the hardware bridge. This must be abstracted in the higher layer with the use of the PON adapter interface and the specific lower layer functionality implemented in the corresponding PON adapter module. All PON adapter modules are independent of each other.

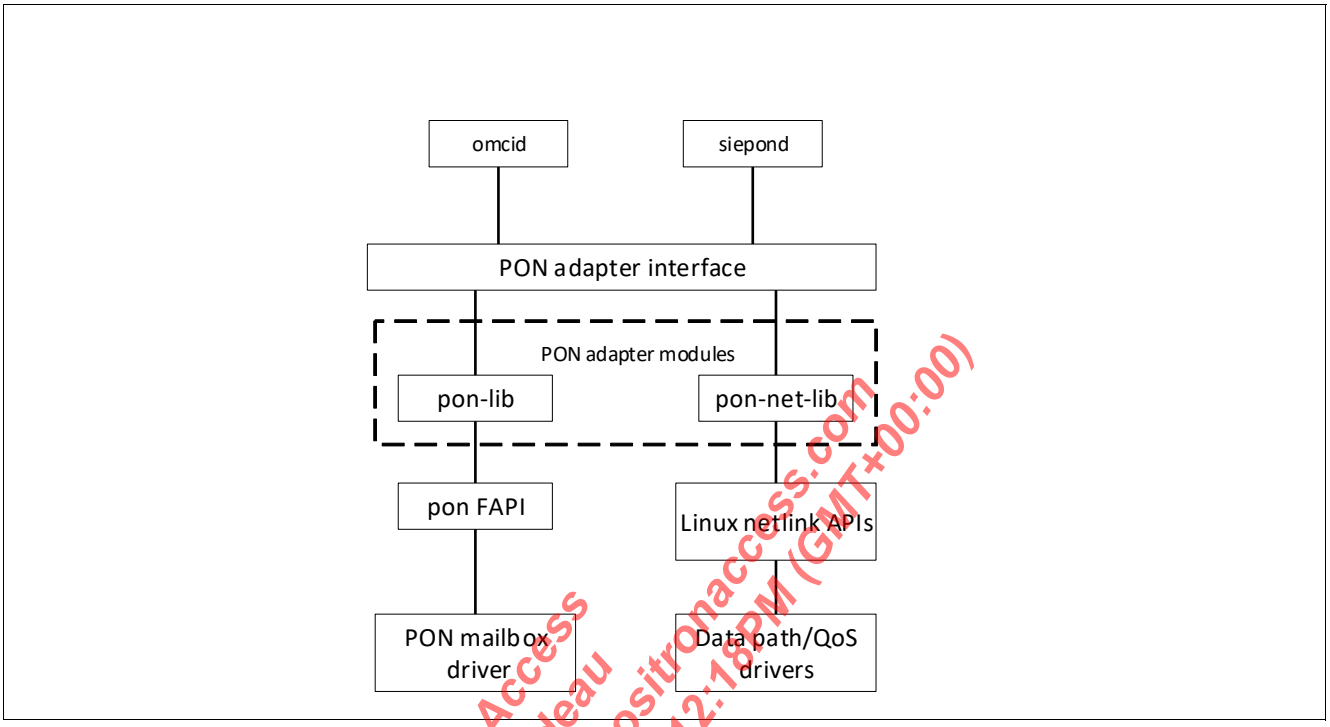


Figure 131 PON Adapter Software Structure

8.3 PON Adapter and Multi-threaded Code

The higher layer application layers such as OMCI must take special care when calling functions exported by the PON libraries.

The libraries are able to do shared resource management and the bookkeeping required for the correct configuration of the system data path. It is not possible to call a function in the libraries from different threads as it may lead to unexpected results. The libraries store information about the related managed entities. The relationship between the managed entities is important for the correct data path configuration. When different threads call a function to configure different entities, this may produce incorrect or unexpected results.

Nevertheless, it is possible to use different threads but the higher layer must implement the correct locking between the threads to ensure the library-internal data is in sync.

This is a summary of thread safety guarantees of the different libraries:

- `pon_adapter`
  This is an interface implementation so no special handling is required to guarantee thread safety.
- `pon_lib`
  The higher layer must implement a thread locking mechanism for each `pon_lib` adapter layer call.
  For the PON FAPI calls, it is possible for each thread to get its own FAPI context and use it without locking.
- `pon_net_lib`
  The higher layer must implement a thread locking mechanism for each `pon_net_lib` adapter call.

9 Frequency and Time Synchronization

Several mechanisms are provided for time, phase, and frequency synchronization of output signals to a selectable input signal:

- Physical layer frequency synchronization (Synchronous Ethernet), see **Section 9.1**
- PON Time-of-Day synchronization, see **Section 9.2**
- Packet based time/phase synchronization (Precision Time Protocol PTP), see **Section 9.3**
- General-purpose Time-of-Day (ToD)/1PPS time/phase synchronization

Figure 132 depicts an overview.

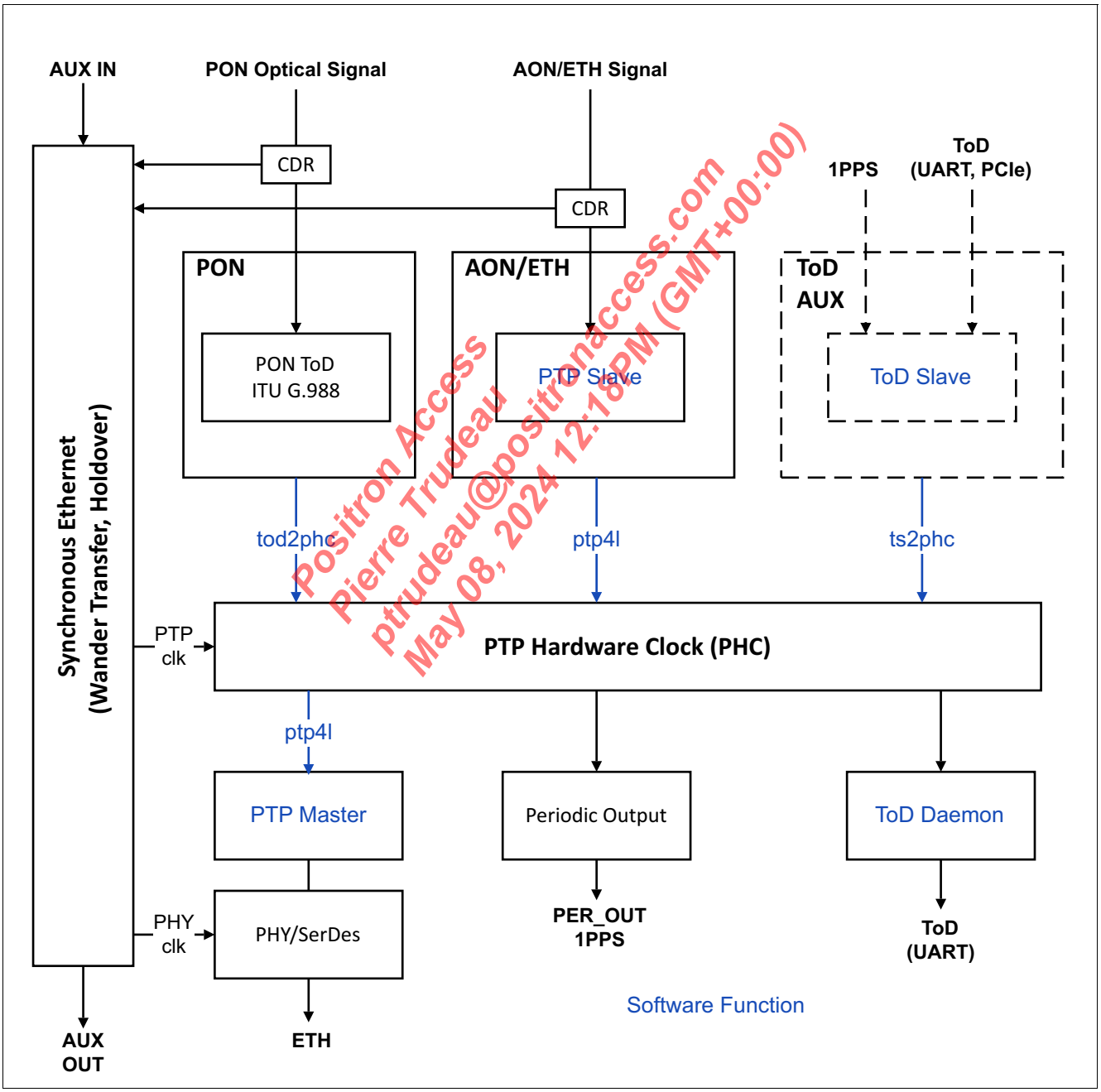


Figure 132 Time and Clock Synchronization Options

The central part for time/phase synchronization is the PTP hardware clock (PHC). It provides the time and phase reference for packet timestamping and timing output signals.

The PHC can be synchronized to:

- PON ToD by usage of the user-space daemon `tod2phc`, which adjusts the PHC to PON ToD.
- AON/PTP slave port by usage of the user-space daemon `ptp4l`, which adjusts the PHC to the PTP slave port.
- A general-purpose ToD/1PPS interface, which allows time/phase synchronization to an external ToD value over various physical interfaces, such as UART, PCIe, or I2C[1)].

Frequency, Time, and Phase output can be provided as:

- Physical Layer (SyncE) output via Ethernet UNI ports, and/or auxiliary output clock signals
- PTP master on Ethernet ports by usage of the `ptp4l` daemon
- 1PPS and serial ToD (UART) output with various ITU and NMEA formats
- Phase-synchronous periodic output clock with programmable interval

1) The synchronization to external timing sources through general-purpose ToD/1PPS requires application specific SW/drivers that are not part of the SDK.

9.1 Synchronous Ethernet

This section describes the implementation and operation of the synchronous Ethernet (SyncE) functionality for PON, AON, and Ethernet applications.

9.1.1 Overview

The functionality of SyncE comprises:

- DPLL for ITU-T G.8262 Ethernet Embedded Clock (EEC) option 1 and ITU-T G.8262.1 enhanced Synchronous Ethernet (eEEC) wander transfer and holdover functionality.
- Selectable SyncE frequency sources
  - WAN, LAN0, LAN1, recovered clock from all supported ETH and PON modes
  - GPHY recovered clock (1000BASE-T and 2.5GBASE-T)
  - Auxiliary external inputs (50 MHz, 25 MHz, 10 MHz, 2.048 MHz, or 1.544 MHz)
- ITU-T G.8262/G.8262.1 compliant short term holdover (15 s) with a regular crystal oscillator (XTAL).
- ITU-T G.8262/G.8262.1 compliant long term holdover and free-run accuracy with an external 4.6 ppm temperature compensated crystal oscillator (TCXO)
- Various SyncE output signals
  - WAN, LAN0, LAN1 transmit signals with all supported Ethernet and PON modes
  - GPHY transmit signal (1000BASE-T and 2.5GBASE-T)
  - Auxiliary output frequencies (25 MHz, 10 MHz, 2.048 MHz, 1.544 MHz, or 8 kHz)
- Reference clock (500 MHz) for frequency-assisted PTP, as required for ITU-T G.8273.2 T-BC
- API for SyncE configuration and status
- Optional input/output for looping through an external SyncE/DPLL device

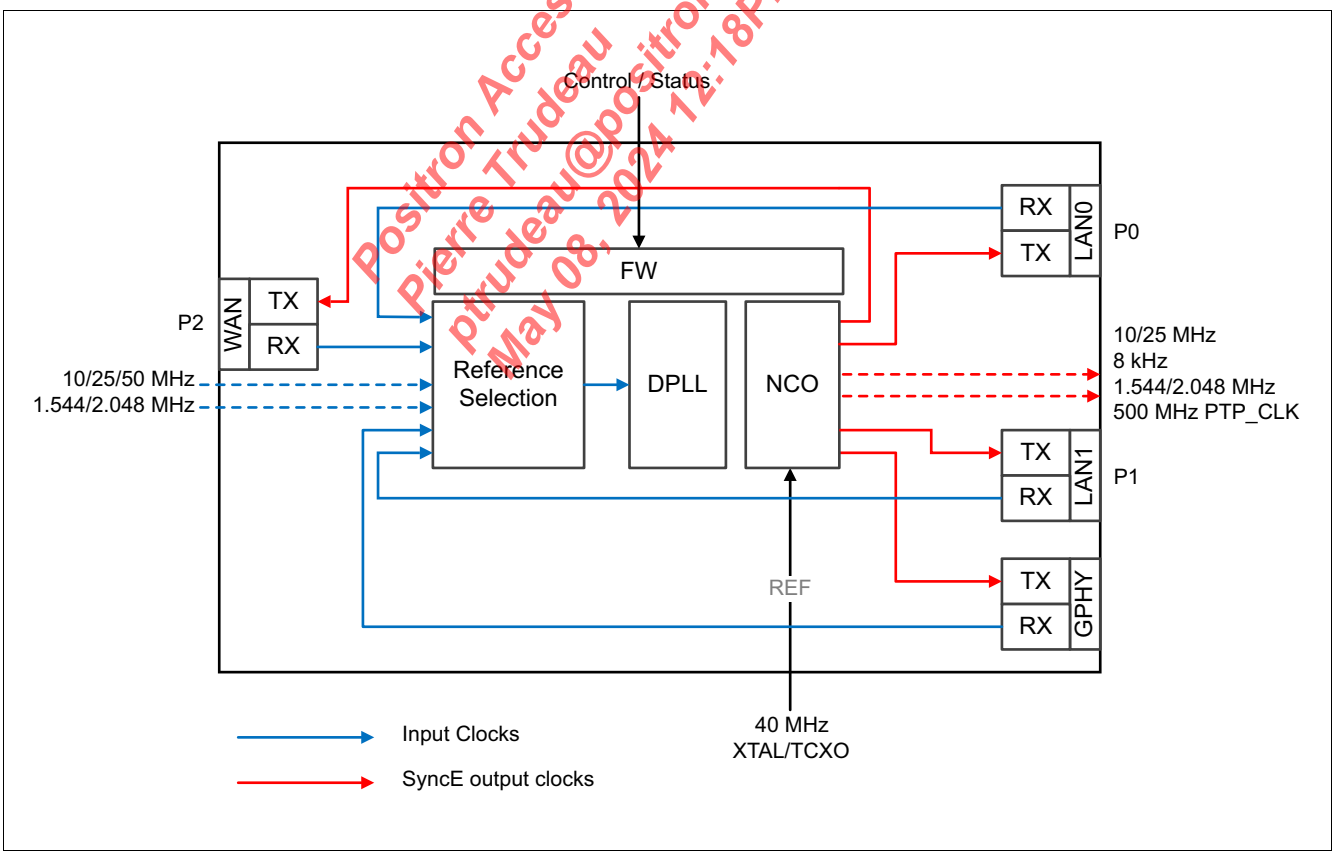


Figure 133 ITU-T G.8262 SyncE Node

9.1.2 Synchronous Ethernet API Commands

The synchronous Ethernet operation mode is controlled by the PON IP hardware and firmware.

The SyncE API comprises these commands:

- `fapi_pon_synce_cfg_set`
  Table 77 shows the configuration parameter details.
- `fapi_pon_synce_cfg_get`
- `fapi_pon_synce_enable`
  This command starts SyncE operation according to the configuration done per synce_cfg_set. If the given SyncE source is not valid (for example out of frequency range or with loss-of-signal indication), the SyncE functionality will remain disabled.
- `fapi_pon_synce_disable`
  This command disables SyncE function. The SyncE output signal returns to the fixed nominal frequency (controlled by the XTAL reference clock).
- `fapi_pon_synce_hold_enable`
  This command forces the SyncE function to enter the holdover state. The SyncE output signal is kept at the last average value. It is only subject to frequency variations that are caused by the clock reference (XTAL, TCXO).
- `fapi_pon_synce_hold_disable`
  This command terminates the SyncE holdover state, if the holdover mode was entered either by the `fapi_pon_synce_hold_enable` command or if the previously selected SyncE source became invalid and the KEEP_HOLD parameter was set to KEEP. The SyncE output signal synchronizes to the selected source. The source can differ from the previously selected source, if changed with the `fapi_pon_synce_config_set` command.
- `fapi_pon_synce_status_get`
  This command returns the current SyncE status.
  0: OFF
  1: SYNCING
  2: SYNC
  3: HOLD

Table 77 Synchronous Ethernet Configuration Parameters

PON Library Parameters	Default Value	Description
synce_mode	1	**Synchronous Ethernet Operation Mode** This selects the synchronous Ethernet operation mode. 0_D **BW0** Reserved, do no use. 1_D **BW1** 2.9 Hz (EEC1 and enhanced EEC) 2_D **BW2** Reserved 3_D **BW3** Reserved
loop_mode	1	**Loop Mode** This selects the synchronous Ethernet loop timing operation mode. 0_D **LM0** Reserved 1_D **FW** DPLL and holdover in FW (default)

Table 77 Synchronous Ethernet Configuration Parameters (cont'd)

PON Library Parameters	Default Value	Description
src	2	**Synchronous Ethernet Source Selection** This selects the synchronous Ethernet synchronization source interface. 0_D **P0** The LAN0 interface recovered receive clock 1_D **P1** The LAN1 interface recovered receive clock 2_D **P2** The WAN interface recovered receive clock 3_D **GPHY** The GPHY recovered receive clock 4_D **GPC1** The general purpose clock input pin GPC1 5_D **GPC2** The general purpose clock input pin GPC2
frq	11	**Synchronous Ethernet Reference Frequency** This selects the input frequency to the DPLL. 0_D **DS1** 1.544 MHz (external reference clock, such as BITS) 1_D **E1** 2.048 MHz (external reference clock, such as BITS) 2_D **F10M** 10.0 MHz (external reference clock, such as GPS) 3_D **F25M** 25.0 MHz (external reference clock, general purpose) 4_D **F50M** 50.0 MHz (external reference clock, general purpose) 5_D **RES1** Reserved, do not use. 6_D **F125M** 125 MHz, this is used for 1000BASE-X/SGMII. 7_D **F155M** 155.52 MHz (G-PON) 8_D **RES2** Reserved, do not use. 9_D **RES3** Reserved, do not use. 10_D **F312M** 312.5 MHz, this is used for 10GBASE-R. 11_D **F622M** 622.08 MHz (XGS-PON, XG-PON, NG-PON2)
n0_avg	6	**Holdover Average Coefficient N0** This value configures the firmware holdover averaging algorithm in units of 2^(N0+8) samples of 125 µs each. The valid range is from 0 to 15, resulting in an averaging interval from 32 ms to 17 min. The default value is 6, corresponding to an averaging interval of 2 seconds.
iir		**IIR Coefficient** This value configures the firmware IIR algorithm. Do not change this value. iir(31:0) = 0x445544dd
ddt	32	**Divider Deviance Threshold** Defines maximum allowed deviance of a DPLL divider value from its predecessor. This can be used as an input phase drift/wander detector. Exceeding this threshold leads to hold-over.
lim_thr	0	**Limiter Threshold** The maximum output frequency difference (w.r.t. XTAL/TCXO) tolerated. The value is given in units of 0.1 ppm. The maximum accepted value is 9960 (996 ppm), typically used values are far below. A value of 1000 corresponds to a frequency range of +/- 100 ppm. A value of 0 disables the limiter function.

Usage:

```
# pon synce_cfg_set
Long Form: synce_cfg_set
Short Form: scs
Input Parameter
-uint8_t synce_mode
-uint8_t loop_mode
-uint8_t src
-uint8_t frq
-uint32_t n0_avg
-uint32_t ddt
-uint32_t lim_thr
-uint32_t iir
```

Configuration example for XGS-PON input on WAN interface and 2.9 Hz bandwidth, Output Limiter Off

```
# pon scs 1 1 2 11 6 32 0 1146438877
```

Configuration example for 10GBASE-R input on WAN and 2.9 Hz bandwidth, Output Limiter +/-50 ppm

```
# pon scs 1 1 2 10 6 32 500 1146438877
```

9.1.3 PHY Master Mode Operation

To enable Synchronous Ethernet, the PHY interface must be run in master mode. This is selected in the driver located at linux/browse/drivers/net/phy/lantiq.c.

```
val = phy_read(phydev, MII_GCTRL);
if (val < 0)
        return val;
val |= MII_GCTRL_MS;
val |= MII_GCTRL_MSEN;
err = phy_write(phydev, MII_GCTRL, val);
```

Please note that two device interfaces that have the master mode set cannot be connected back-to-back. To enable this, the above code must be removed for one of the devices/interfaces.

9.2 PON Time of Day

This chapter describes the configuration and operation of the ITU-T PON ToD mechanism which is controlled by the OLT through the OMCI OLT-G managed entity. This provides a mechanism to transport the Time of Day (ToD) information from the OLT to the ONU. This involves the PON IP hardware, the PON firmware, and the OMCI software.

The PON IP hardware synchronizes the ToD to a pre-programmed multiframe count configured through an OLT-G attribute. Once the synchronization has happened, the PON hardware is aware of the actual time of day. Latencies and delays due to fiber length, as well as time inaccuracies due to fiber latency asymmetry are already corrected by the OLT.

When synchronized, the PON IP hardware drives the 1PPS signal, which is used internally for PTP hardware clock (PHC) synchronization. If an external 1PPS output signal is required, this is taken from the PHC.

PON ToD Configuration

The PON ToD configuration, including parameters for OLT-G synchronization, is defined in the UCI configuration file `/etc/config/tod`, as described in **Chapter 2.4.3.12, UCI Configuration – ToD Timing**.

The values provided in this configuration file have been tested against a specific OLT model. For operation with other OLT models, modify these parameters as required.

Accessing PON ToD

The PON ToD is accessible in two ways:

- PON library functions `fapi_pon_gpon_tod_sync_set/_get`
- Linux PTP API `/dev/ptp1, /dev/ptp2`
  Examples:
  ```
  phc_ctl /dev/ptp1 get
  testptp -d /dev/ptp2 -g
  ```

Only read operations (`gettime`) and captured time on 1PPS or AUX_IN events (`extts`) are supported through the PTP API. Writing or adjusting PON ToD through the PTP API is prohibited, because PON ToD is under control of the OMCI ToD mechanism.
Two independent PTP devices are provided by the PON PTP driver. This is important, so that each `extts` event is associated with a separate kernel event queue for independent reading by the user space applications. The `extts` timestamp read by `phc2sys` from `/dev/ptp1` is used to synchronize the PHC to PON ToD. The `extts` timestamp read from `/dev/ptp2` is readable by the ToD daemon (see **Chapter 9.2**).

If the PON ToD time is invalid, or not synchronized yet, the timestamp reported by `extts` indicates this special condition:

- ToD invalid, SyncE = on: extts = 0xFFFFFFFFFFFFFFFF
- ToD invalid, SyncE = off: extts = 0xFFFFFFFFFFFFFFFE
- For all other conditions with ToD valid: `extts`, the timestamp holds the 64-bit ToD value at the time of the PON 1PPS event.

9.3 Precision Timing Protocol

This section describes the functions implemented for the Precision Timing Protocol (PTP, IEEE 1588v2).

9.3.1 Relation to Standards Requirements

IEEE 1588-2008 (1588v2) defines a framework and mechanism for packet-based time and frequency synchronization. The IEEE 1588 functionality described in this document refers to the telecom clock according to these ITU-T standards:

- Telecom boundary clock ITU-T G.8273.2 **[61]**
- Telecom profile ITU-T G.8275.1 **[63]**
- Telecom profile ITU-T G.8275.2 **[64]**

9.3.2 Features

These are the PTP features:

- PON:
  IEEE1588-2008 (1588v2) MOOC (Master Only Ordinary Clock) supporting distributed ITU-T G.8273.2 T-BC, Class C
- AON/ETH:
  IEEE1588-2008 (1588v2), ITU-T G.8273.2 T-BC, Class C
- 1-step and 2-step time stamping
- ITU-T G.8275.1 PTP telecom profile for time and phase synchronization with full timing support
- ITU-T G.8275.2 PTP telecom profile for time and phase synchronization with partial timing support.
  This includes UDP checksum recalculation for IPv6.
- Comprehensive PTP management API supporting local and remote PTP software stack management
- Frequency-Assisted PTP (Synchronous Ethernet)

9.3.3 Functional Description

Figure 134 shows a detailed overview of the hardware, driver, and software functions for PTP, ToD, 1PPS, and SyncE.

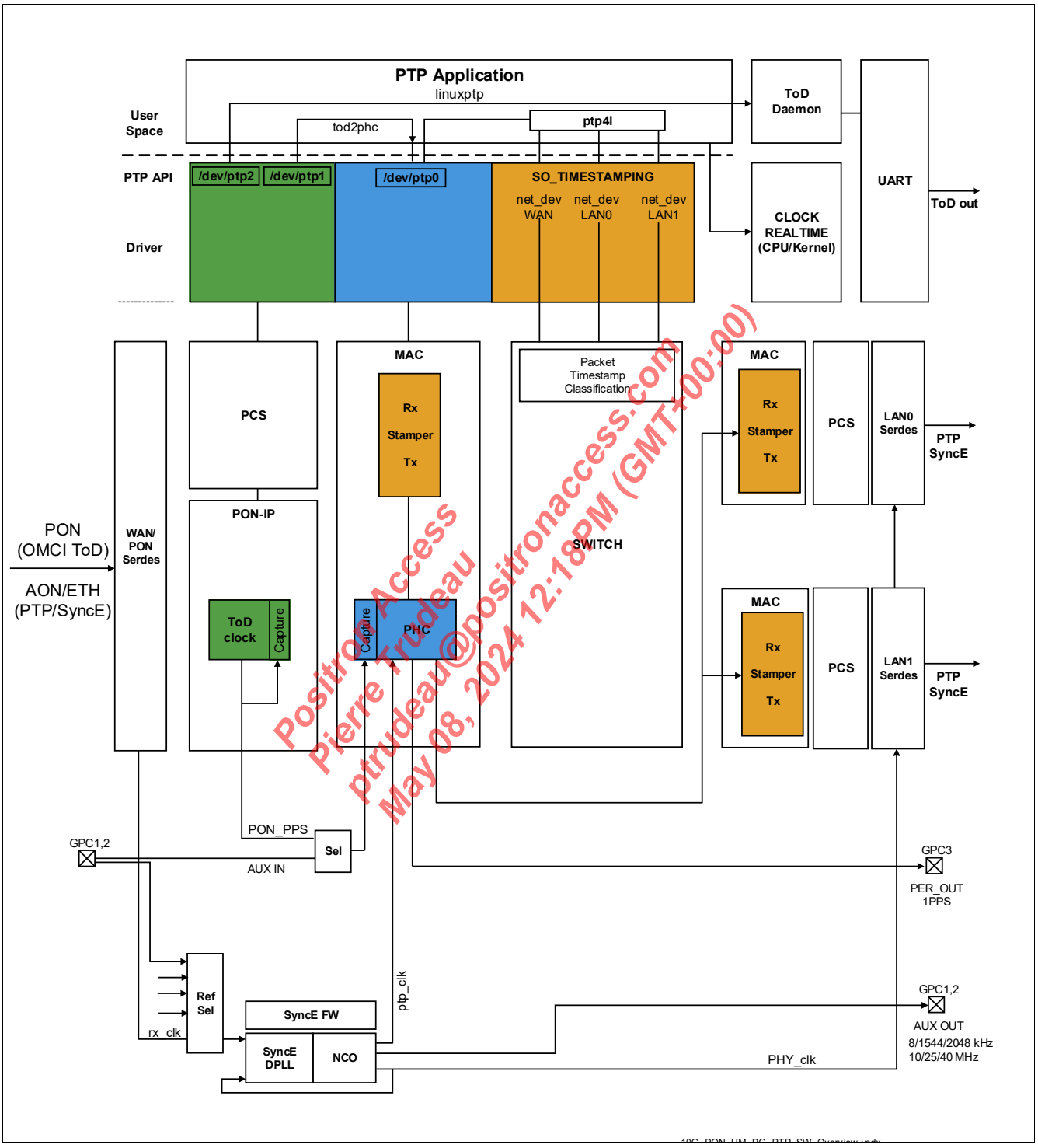


Figure 134 Frequency, Time, and Phase Synchronization Details

9.3.4 PTP Operation Modes

When PTP is operated in full boundary clock mode, each Ethernet port is either PTP master or PTP slave, depending on announce messages and the PTP best master clock algorithm (BMCA) as implemented in `ptp4l`. Ports can be also forced to either master or slave through the PTP configuration file. In boundary clock mode, at least two ports have to be given to the ptp4l start parameters. The PHC is synchronized to the selected slave PTP port. **Figure 135** shows the PTP packet flow.

When PTP is operated in ordinary clock mode, the Ethernet port is usually configured as PTP master port (Master Only Ordinary Clock, MOOC). When in MOOC mode, the PHC must be synchronized to an alternate time source (for example the PON ToD). **Figure 136** shows the PTP packet flow for two master ports synchronized to PON ToD.

In both operation modes, physical layer frequency assistance (SyncE) is available, as described in **Section 9.3.5**.

9.3.4.1 Boundary Clock in ETH/AON Context

In Ethernet/AON mode, all Ethernet ports can participate in the BMCA (Best Master Clock Algorithm) of `ptp4l`.

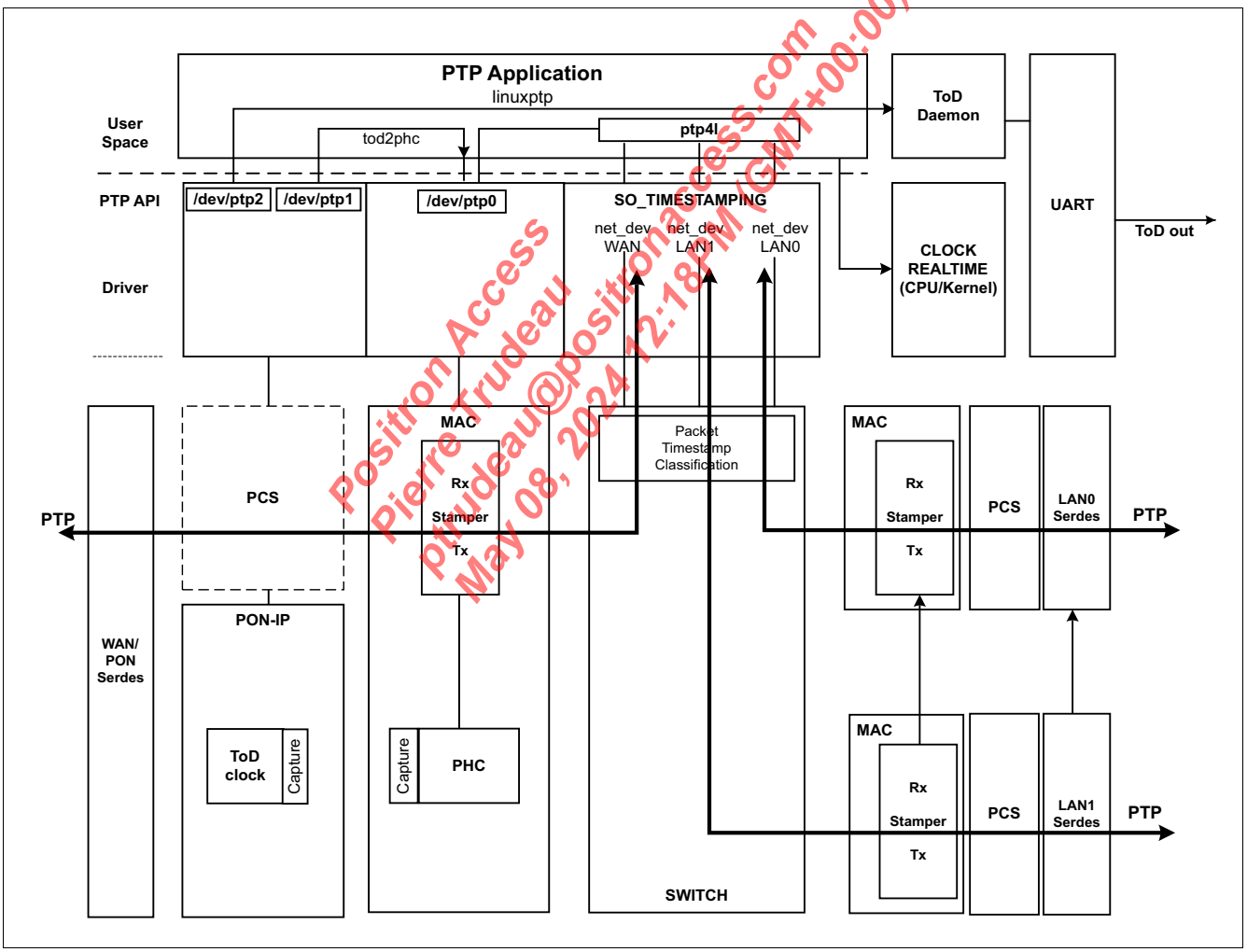


Figure 135 Traffic Flow of T-BC in ETH/AON Context

9.3.4.2 Boundary Clock in PON Context

In the context of PON, the boundary clock node is seen as a distributed node. In this distributed clock model, the T-BC slave is implemented within the OLT as a Slave Only Ordinary Clock (SOOC), the T-BC master is implemented within the ONU as a Master Only Ordinary Clock (MOOC).

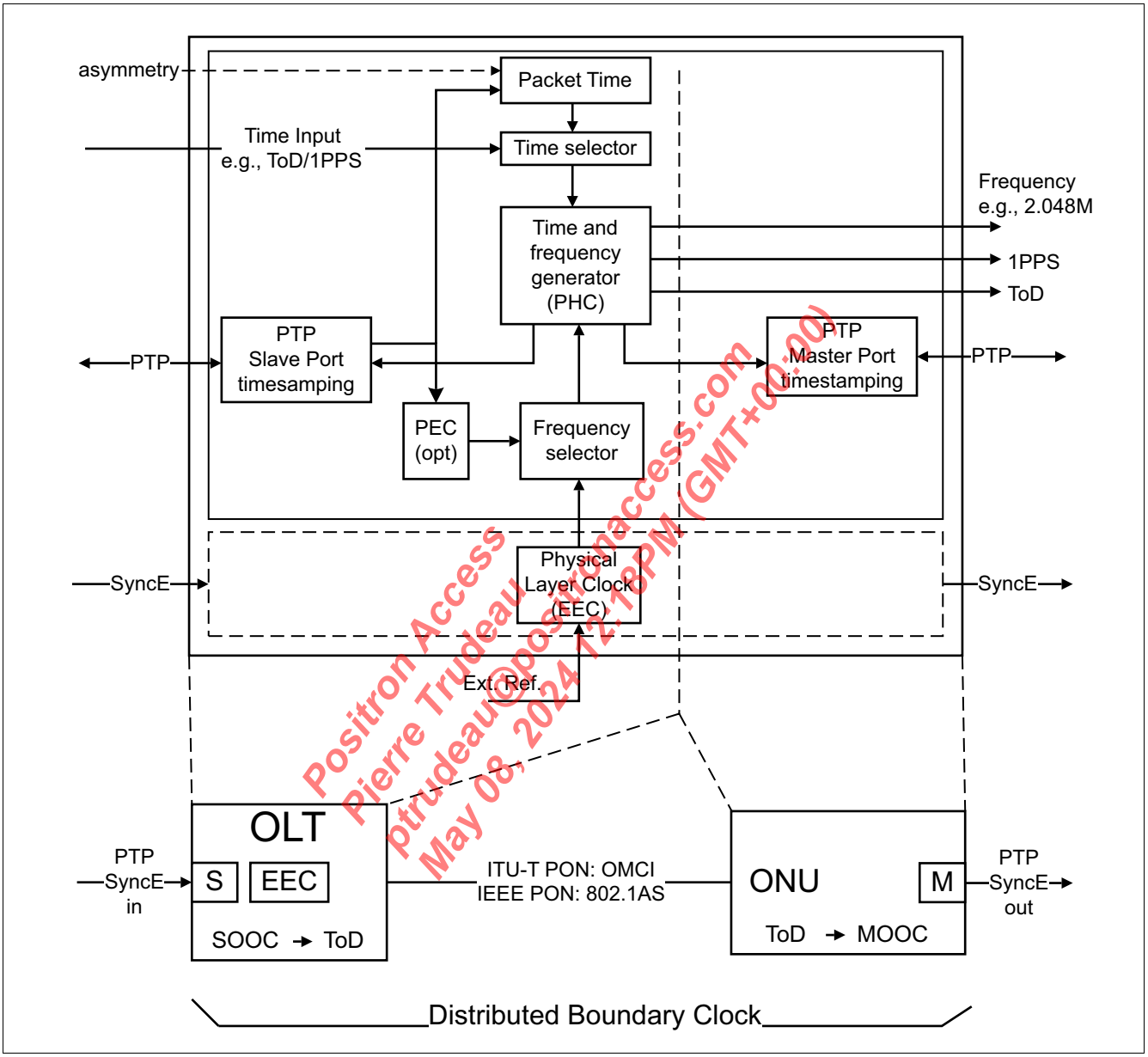


Figure 136 ITU-T G.8273.2 Clock Model in PON Context

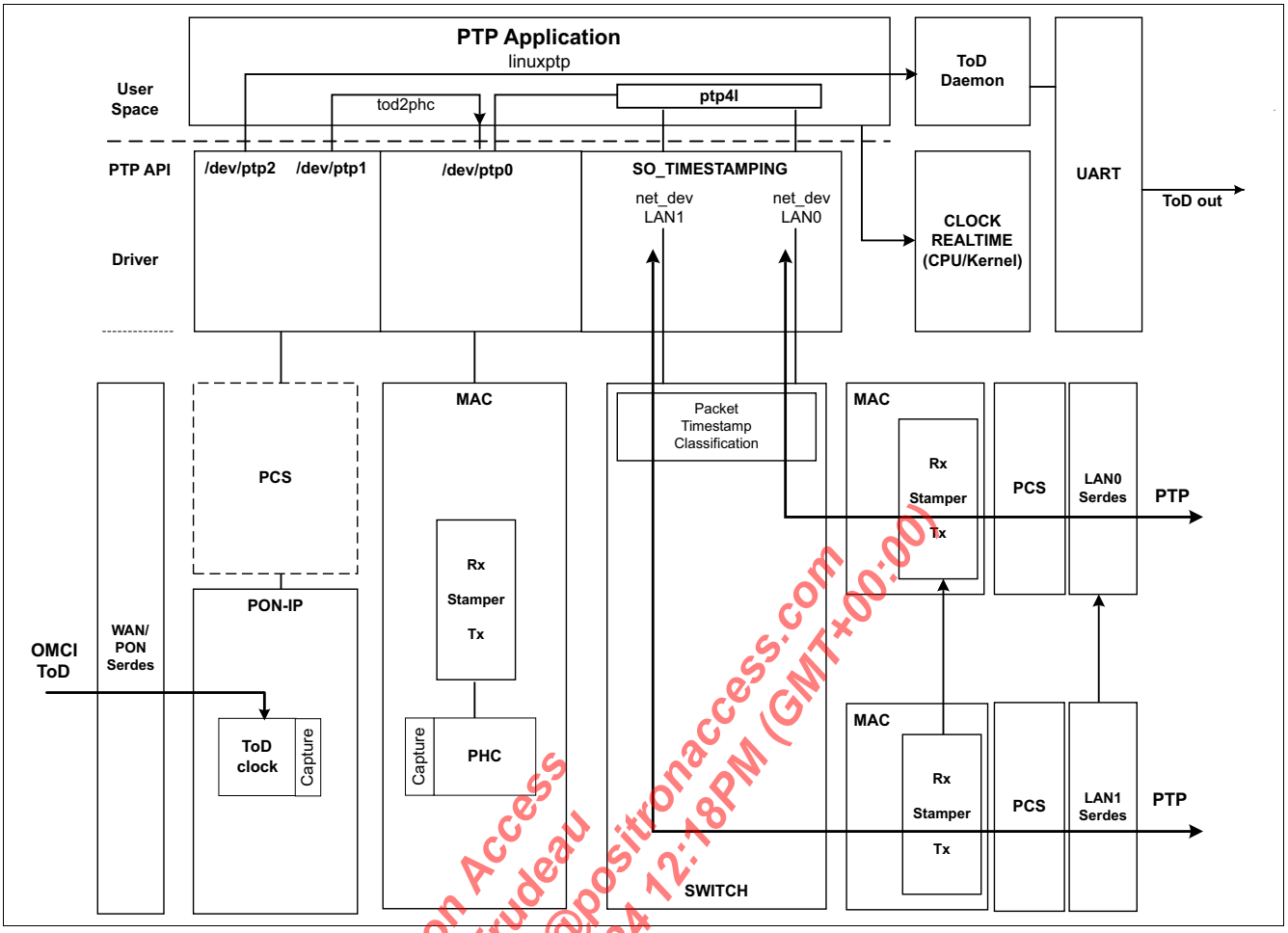


Figure 137 Traffic Flow of PTP Master (MOOC) in PON Context

9.3.5 Frequency Assisted PTP

There are two options to control the frequency and phase of the PHC:

- PTP clock control API through `/dev/ptp*`
  The increment rate of the PHC is digitally controlled through the `adjfreq()` function of the PTP API.
  When ptp4l is used for PHC control, the PHC follows the PTP-to-PTP noise transfer function as defined in **[61]**.
- Physical clock ptp_clk as shown in **Figure 134**.
  When SyncE is enabled, this clock is synchronous to the SyncE output. It follows the SyncE noise transfer function and holdover modes. This corresponds to the SyncE-to-PTP noise transfer function as defined in **[61]**.

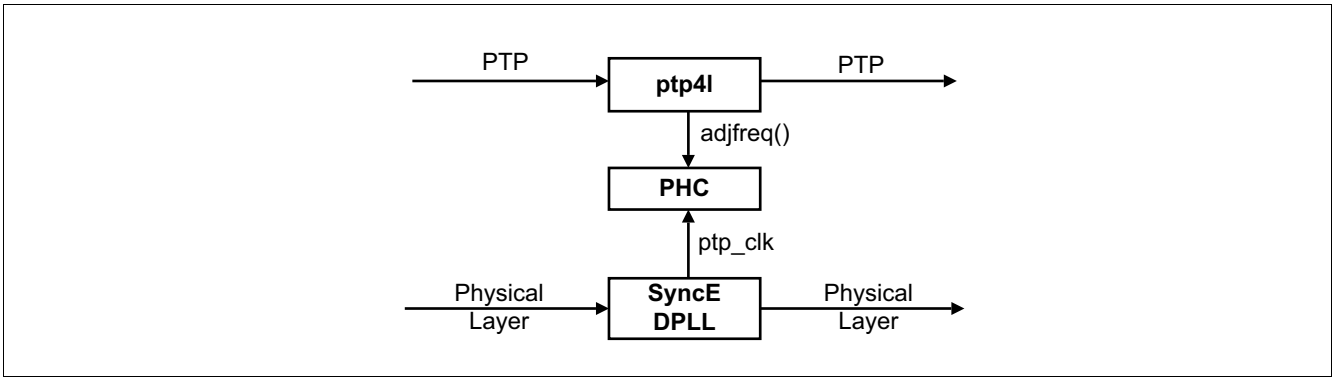


Figure 138 Frequency Assisted PTP

9.3.6 PTP Clock Control API

The PTP driver registers a certain number of `/dev/ptp` devices.
The PTP device provides an API for reading, writing, and adjusting the PHC of the associated PTP device.

- `gettime()`: arbitrary read of the PHC time
- `settime()`: arbitrary write of the PHC time
- `adjtime()`: adjust the time in nanoseconds
- `adjfreq()`: adjust the frequency/increment rate in ppb (parts per billion)
- `extts()` : captured value of the PHC time on external event (for example 1PPS)
- `per_out()`: configuration of the periodic output (for example 1PPS or 50 Hz)

If more than one application is associated with an `extts` timestamp, the driver must provide an additional PTP device for a separate `extts` event queue. This is the case for example with PON ToD, where two applications require access to the `extts`:

`/dev/ptp0`
PHC is used to timestamp all PTP packets on all Ethernet ports.

- In PON mode, it is used to timestamp PTP packets on UNI/LAN ports, the PHC is controlled by `tod2phc`.
- In AON/ETH mode, it is used to timestamp PTP packets on WAN and UNI/LAN ports, the PHC is controlled by `ptp4l`.

`/dev/ptp1`
This is the PON TOD event queue that is usually used as source clock for `tod2phc`.

`/dev/ptp2`
This is the PON TOD event queue that is usually used as source clock for TOD daemon `todd`.

9.3.7 Socket Control Interface SO_TIMESTAMPING

`SO_TIMESTAMPING` is an interface for receiving network packet timestamps. It generates timestamps on reception, transmission, or both. It supports multiple timestamp sources, including hardware time stamping. The PTP driver provides this socket control interface for hardware time stamping on the associated interface.

9.3.8 Linux PTP Protocol Stack

This software is an implementation of the precision time protocol (PTP) according to IEEE standard 1588 for Linux. For details, refer to **http://linuxptp.sourceforge.net/** and **http://sourceforge.net/projects/linuxptp/**.

9.3.8.1 License

The software is copyrighted by the authors and is licensed under the GNU general public license.

9.3.8.2 Features

The Linux PTP software provides these features.

- Supports hardware and software time stamping via the `SO_TIMESTAMPING` socket option.
- Supports the Linux PTP Hardware Clock (PHC) subsystem.
- Implements Boundary Clock (BC) and Ordinary Clock (OC).
- Transport over UDP/IPv4, UDP/IPv6, and raw Ethernet (Layer 2).

9.3.8.3 User Space Applications

Linux PTP provides three main user applications.

9.3.8.3.1 phc2sys

This is a program which synchronizes two clocks in the system. Normally, it is used to synchronize the system clock (kernel timer) to a PTP hardware clock (PHC). It can be also used to synchronize two hardware clocks, which are the PON ToD and PHC. For details, refer to **http://linux.die.net/man/8/phc2sys**.

The original version of the user space application `phc2sys` performs an arbitrary timer read of the source and destination clocks by `gettime()` and calculates the required time or frequency adjustment of the slave clock. Using `gettime()` leads to a sequential reading of the two clocks, which does not allow a precise comparison.

Therefore, there is an additional option added to the original version. The -e option reads the source and destination clocks as captured on the external 1PPS event using the `extts()` function of the PTP API. This allows an exact comparison of both clocks. The slave clock is adjusted to minimize the time error between both clocks. As both clocks have identical increment rates (SyncE), the slave adjustment is done only once, or at any time the source clock (PON ToD) obtains a new value via the ITU-T G.984.3amd2 mechanism.

In any case, it is assumed that the source and destination clocks are always kept synchronous with minimal time difference (of about 4-5 ns, according to the resolution of the `extts` time stamps).

Example 1

Using `phc2sys` to adjust the kernel system time to PHC:

```
phc2sys -s /dev/ptp0 -c CLOCK_REALTIME -O 0 -S 0.00002 -m -q
/dev/ptp0: Master PHC
CLOCK_REATLIME: the kernel system clock
```

Example 2

Using to adjust PHC to PON ToD with the -e option.

```
phc2sys -s /dev/ptp1 -e 0 -c /dev/ptp0 -e 0 -O 0 -T 20 -o -14 -l 6 -S 0.00002

/dev/ptp1: Master's PHC (PON ToD)

/dev/ptp0: Slave's PHC
```

```
Option -o (lower case): Offset (14 nanoseconds) between PON ToD and PHC, to
compensate internal latencies.

Option -S: Specify the step threshold of the servo. It is the maximum offset that
the servo corrects by changing the clock frequency instead of stepping the clock.
With ToD adjustments (from the OLT) exceeding 20 µsec, the PHC clock will be stepped
for faster alignment to large ToD adjustments. Note that these large ToD adjustments
from OLT are an exception and against the requirement of monotonous incrementing PTP
wall clocks (PHC).
```

When the option -e is used, phc2sys operates based on these rules:

- If extts timestamp = 0xFFFFFFFFFFFFFFFF (SyncE = on, extts source timestamp = invalid) adjfreq = 0. The PHC frequency will rely on SyncE only.
- If extts timestamp = 0xFFFFFFFFFFFFFFFE (SyncE = off, extts source timestamp = invalid) no updates to adjfreq and adjtime, PHC continues with last known good frequency.
- Else: perform normal synchronization of the given clocks.

9.3.8.3.2 ptp4l

This is an implementation of the PTP according to IEEE standard 1588 for Linux. It implements BC and OC.

Ordinary Clock Operation (MOOC)

The operation mode (clockType) of ptp4l is automatically detected by the number of available ports. In this ONU SFP application, there is only 1 port available, and ptp4l operates with `clockType = CLOCK_TYPE_ORDINARY`.

Example

```
ptp4l -E -2 -H -f /etc/g_8275_1.conf -i eth0_0 -m -q
-E: select the E2E mechanism
-2: use layer 2 802.3 (Raw Ethernet)
-H: use HW time stamping
-f: use the given configuration file
-i: use lct0 (the netdev of the SW exception queue for 1588)
```

For details, refer to **http://linux.die.net/man/8/ptp4l**.

9.3.8.3.3 pmc

This is a program which implements a PTP management client according to IEEE standard 1588. The program translates input actions specified by name and management ID into management messages, sends them over the selected transport protocol, and returns any received replies.

There are three actions supported:

- GET retrieves the specified information,
- SET updates the specified information, and
- CMD (or COMMAND) initiates the specified event.

For details, refer to **http://linux.die.net/man/8/pmc**.

9.3.9 Linux PTP Configuration, Operation, and Management

For PTP configuration and operation, the configuration files shown in **Table 78** and scripts as described in **Chapter 9.3.9.1** and **Chapter 9.3.9.2** are provided.

Table 78 IEEE1588 Configuration Files and Scripts

File	Description
`/usr/bin/ptp_sync.sh`	Main start/stop script for ptp4l (includes `tod2phc`). This script performs all steps required for ptp4l activation.
`/usr/bin/tod2phc.sh`	Start/stop script to synchronize the GSW1xx timer to ToD. This script is used to start GSW1xx synchronization without starting ptp4l.
`/etc/config/ieee_1588`	Start script options for `tod2phc.sh` and `ptp_sync.sh`.
`/etc/g_8275_1.conf`	ptp4l configuration file with default options according to the ITU-T G.8275.1 telecom profile.

Attention: The /dev/ptp device numbering depends on the number of active LAN interfaces and the driver device registering sequence. Modify the scripts accordingly.

9.3.9.1 ptp_sync.sh

This script is provided to start ptp4l with all relevant options, including all steps defined in `tod2phc.sh`.

`/usr/bin/ptp start/stop/restart`

- Start
  - Checks whether `phc2sys` is already running with option `-e`. When not running, it starts `phc2sys -e` and waits for two seconds until the PHC PTP timer is synchronized to PON ToD.
  - Starts ptp4l with options from `/etc/config/ieee_1588` and default configuration from `/etc/g_8275_1.conf`.
- Stop
  - Stops ptp4l.
  - Stops the PTP event message classification.
  - Stops `phc2sys -e`.
- Restart
  - Stops ptp4l as above.
  - Starts ptp4l as above.

9.3.9.2 tod2phc.sh

A mandatory prerequisite for the usage of a PTP daemon is the precise synchronization of the PHC PTP timer to the PON ToD. The script `/usr/bin/tod2phc.sh` checks for ONU SyncE status reports successfully synchronized, and for valid ONU ToD synchronization after receiving the OLT-G managed entity attributes. The script invokes `phc2sys -e` with all relevant options from `/etc/config/ieee_1588`. `tod2phc.sh` is invoked automatically by `ptp_sync.sh`.

- Start
  - Checks whether PON 1PPS is used as trigger for PHC EXTTS (`switch_cli gswss get aux_trig`).
  - Starts `phc2sys` with the options from `/etc/config/ieee_1588`.
- Stop
  - Stops `phc2sys`.
- Restart
  - Stops as above.
  - Starts as above.

The option `-o` (lower case) provides a mean to compensate for any internal latencies within the synchronization path from PON ToD. This internal time error is compensated by adding an offset with the `-o` option (to be edited in `/etc/config/ieee_1588`). With a properly calibrated compensation, the constant time error (cTE) must be well below 10 ns for support of G.8273.2 T-BC class C. We recommend to not alter the offset as provided in `/etc/config/ieee_1588`. The script `tod2phc.sh` is automatically executed from `ptp_sync.sh`, when required.

9.3.9.3 PTP Management Client

The PTP stack management is performed by management TLVs. They are generated by user proprietary clients, or by using the PTP management client as provided by the `linuxptp` package.

```
usage: pmc [options] [commands]
Network Transport Options
-2      IEEE 802.3
-4      UDP IPv4 (default)
-6      UDP IPv6
-u      UDS local
Other Options
-b []   boundary hops, default 1
-d []   domain number, default 0
-h      prints this message and exits
-i []   interface to use, default eth0 for network
        and /var/run/pmc.$pid for UDS
-s []   server address for UDS, default /var/run/ptp4l
-t []   transport specific field, default 0x0
-v      prints the software version and exits
-z      send zero length TLV values with the GET actions
```

The `[commands]` field has the form `[ACTION MANAGMENT_ID]`

Where `<ACTION>` is either:

```
GET
SET
CMD
```


PMC Local Management

The local ptp4l stack management is provided by running the PMC locally on LCI.

For local communication to the ptp4l, the pmc uses UNIX domain socket (UDS), as indicated by the `-u` option:

```
pmc -u <ACTION MANAGEMENT_ID>
```

Management of Remote PTP Devices

For the management of remote PTP devices (for example a remote PTP slave clock connected to this MOOC), two options are provided:

```
pmc -u -b <hop count>:      for remote management via local UDS.
                            The management TLVs are passed to the local ptp4l via
                            UDS. For accessing further nodes beyond the local node,
                            management TLVs are forwarded via the Ethernet interface
                            associated with the local ptp4l.

                            Management responses are obtained from all nodes with
                            distance up to <hop count> from local node, including
                            the local responses from the local node.

pmc -2 -b <hop count>:      for remote management via Ethernet port.
                            Management responses are obtained from all nodes with
                            distance up to <hop count> from local node, excluding
                            the local node (which is actually bypassed)
```

PMC Management IDs

Example for a pmc command:

```
pmc -u -d 24 -b 1 'GET CURRENT_DATA_SET'
```

Table 79 PMC Clock Management IDs

Management ID	Action	Elements
USER_DESCRIPTION	GET	userDescription
DEFAULT_DATA_SET	GET	twoStepFlag
		slaveOnly
		numberPorts
		priority1
		clockClass
		clockAccuracy
		offsetScaledLogVariance
		priority2
		clockIdentity
		domainNumber
CURRENT_DATA_SET	GET	stepsRemoved
		offsetFromMaster
		meanPathDelay
CURRENT_DATA_SET_STEPS_REMOVED_NP[1)]	GET, SET	stepsRemoved

Table 79 PMC Clock Management IDs (cont'd)

Management ID	Action	Elements
PARENT_DATA_SET	GET	parentPortIdentity
		parentStats
		observedParentOffsetScaledLogVariance
		observedParentClockPhaseChangeRate
		grandmasterPriority1
		gm.ClockClass
		gm.ClockAccuracy
		gm.OffsetScaledLogVariance
		grandmasterPriority2
		grandmasterIdentity
TIME_PROPERTIES_DATA_SET	GET	currentUtcOffset
		leap61
		leap59
		currentUtcOffsetValid
		ptpTimescale
		timeTraceable
		frequencyTraceable
		timeSource
PRIORITY1	GET, SET	priority1
PRIORITY2	GET, SET	priority2
DOMAIN	GET, SET	domainNumber
SLAVE_ONLY	GET	slaveOnly
CLOCK_ACCURACY	GET, SET	clockAccuracy
TRACEABILITY_PROPERTIES	GET, SET	timeTraceable
		frequencyTraceable
TIMESCALE_PROPERTIES	GET, SET	ptpTimescale
TIME_STATUS_NP	GET	master_offset
		ingress_time
		cumulativeScaledRateOffset
		scaledLastGmPhaseChange
		gmTimeBaseIndicator
		lastGmPhaseChange
		gmPresent
		gmIdentity

Table 79 PMC Clock Management IDs (cont'd)

Management ID	Action	Elements
GRANDMASTER_SETTINGS_NP	GET, SET	clockClass
		clockAccuracy
		offsetScaledLogVariance
		currentUtcOffset
		leap61
		leap59
		currentUtcOffsetValid
		ptpTimescale
		timeTraceable
		frequencyTraceable
		timeSource
		externalGrandmasterId As long as not set (default): The local clockID is announced as grandmasterID When SET to xxxxxx.xxxx.xxxxxx: This value is announced as grandmasterID. When SET to 000000.000000.000000 (reset to default): The local clockID is announced as grandmasterID.
SUBSCRIBE_EVENTS_NP	GET, SET	duration
		NOTIFY_PORT_STATE
SYNCHRONIZATION_UNCERTAIN_NP	GET, SET	uncertain

1) NP = Non Portable (not defined in IEEE1588-2008)

Table 80 PMC Port Management IDs

Management ID	Action	Elements
CLOCK_DESCRIPTION	GET	clockType
		physicalLayerProtocol
		physicalAddress
		protocolAddress
		manufacturerId
		productDescription
		revisionData
		userDescription
		profileId

Table 80 PMC Port Management IDs (cont'd)

Management ID	Action	Elements
PORT_DATA_SET	GET	portIdentity
		portState
		logMinDelayReqInterval
		peerMeanPathDelay
		logAnnounceInterval
		announceReceiptTimeout
		logSyncInterval
		delayMechanism
		logMinPdelayReqInterval
		versionNumber
PORT_DATA_SET_NP	GET, SET	neighborPropDelayThresh
		asCapable
STATISTICS_DATA_SET_NP	GET	pmTime (CLOCK_MONOTONIC)
		announceTx
		announceRx
		syncTx
		syncRx
		followUpTx
		followUpRx
		delayReqTx
		delayReqRx
		delayRespTx
		delayRespRx
LOG_ANNOUNCE_INTERVAL	GET, SET	logAnnounceInterval
ANNOUNCE_RECEIPT_TIMEOUT	GET	announceReceiptTimeout
LOG_SYNC_INTERVAL	GET, SET	logSyncInterval
VERSION_NUMBER	GET	versionNumber
DELAY_MECHANISM	GET, SET	delayMechanism
LOG_MIN_PDELAY_REQ_INTERVAL	GET	logMinPdelayReqInterval

Table 80 PMC Port Management IDs (cont'd)

Management ID	Action	Elements
PORT_STATS_NP	GET	rxSync
		rxDelayReq
		rxPdelayReq
		rxPdelayResp
		rxFollowUp
		rxDelayResp
		rxPdelayRespFollowUp
		rxAnnounce
		rxSignaling
		rxManagement
		txSync
		txDelayReq
		txPdelayReq
		txPdelayResp
		txFollowUp
		txDelayResp
		txPdelayRespFollowUp
		txAnnounce
		txSignaling
		txManagement
PORT_PROPERTIES_NP	GET	portIdentity
		portState
		timestamping
		interface

A full list of management IDs is obtained by typing:

```
pmc -u help
```

9.3.9.4 PTP Signaling and Control

This is an example of possible functions for:

- Automatic generation of ESMC (QL = DNU) messages in case of LOS
- Automatic GSW1xx/ToD clock synchronization with `phc2sys`
- Automatic setting of the frequencyTraceable
- Automatic setting of the timeTraceable flag.
- Automatic setting of clockClass, clockAccuracy, currentUtcOffset, and leap flags based on the PTP clock information received from OLT (for example via proprietary OMCI managed entities)
- ToD quality
- SYNCE_STATUS
- LOS ALARM

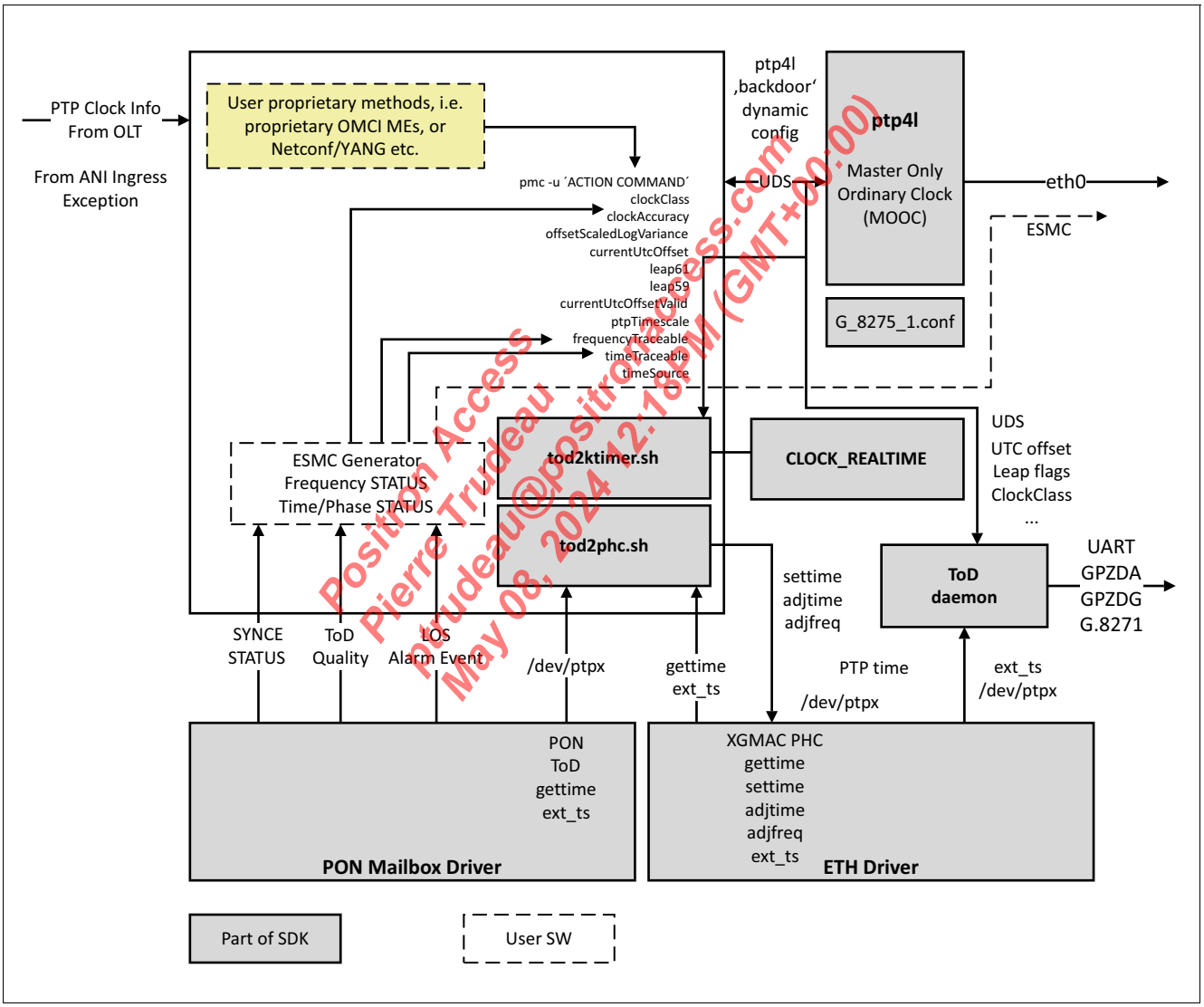


Figure 139 SyncE/PTP Housekeeping

9.4 1PPS and Periodic Clock Output

Note: This description is related to the generic signal generation from the PHC hardware.
This is used from SDK SYS_1.18.0 onwards.
*For earlier versions see **Section 9.6**.*

The signal 1PPS (GPIO29) can be configured to drive a clock synchronization signal. This is a periodic output, synchronous with the PHC. The output signal is configurable in terms of frequency, pulse width and pulse offset.If the PHC is frequency-assisted with SyncE, and if SyncE is in holdover, the 1PPS and Periodic Output will continue with SyncE holdover timing. The overall synchronization signal flow is shown in **Figure 140**.

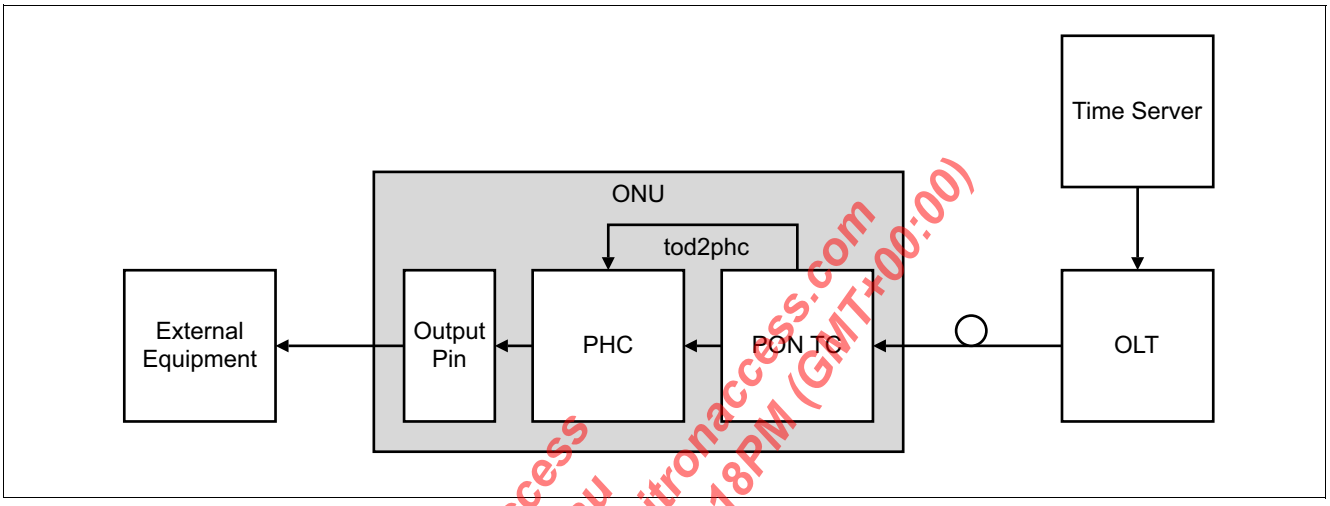


Figure 140 Clock and Time Synchronization Path – PHC Driven

The signal output is configured trough the Linux PTP API, using the PTP_PEROUT_REQUEST2 command.

Set these parameters as required:

- Period in units of seconds and nanoseconds: `period`
  - The value range is from 100 ns to 1 s in steps of 4 ns.
  - The value 0 disables the output.
- Pulse width in units of seconds and nanoseconds: `on`
  - The value range is from 4 ns up to the value of period - 4 ns.
- Phase or start time in units of seconds and nanoseconds: `phase` or `start`
  - The start time is given as a 64-bit value related to the TAI time starting 1970-01-01. See `struct timespec`.

The 1PPS output is deactivated in the DTS by default. To make it usable, either the DTS must be modified before image compilation or, on an existing image, the command `pon pin_config_set 1 2` must be sent.

To further enable the 1PPS signal output, the PON library function `fapi_pon_1pps_event_enable` must be sent once before the first hardware synchronization is configured.

The output function of this pin must be configured in the Linux device tree (using `&pinctrl_gpc3`).

For example:

```
&ponmbox1 {
        status = "okay";
        pinctrl-0 = <&pinctrl_pon_ip &pinctrl_gpc3 &pinctrl_pon_ip_sfp_tx_dis>;
};

&pinctrl {
        pin_1pps_disabled: pin_1pps-disabled {
                intel,groups = "1pps_dis";
                intel,function = "gpio";
                intel,pins = <29>;
                intel,mux = <EQBR_MUX_0>;
        };
};
```

Figure 141 shows the periodic output signal timing derived from the PHC.

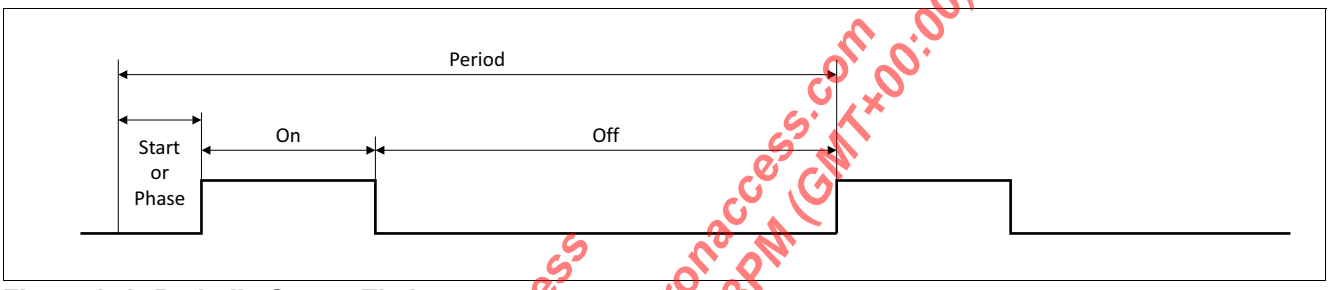


Figure 141 Periodic Output Timing

9.4.1 1PPS and Periodic Output Configuration

`testptp` is a Linux debug tool for the user space PTP API. This can be used to verify the functionality. Some examples are given below. Customer applications are supposed to call the PTP API directly (see `ptp_perout_request`).

Configuration Example: 1PPS Output

This signal format can be used in combination with ToD output of the ToD daemon, see **Section 9.5**.

The pulse width is 50%:

```
testptp -d /dev/ptp0 -p 1000000000 -H 0
```

To disable the output:

```
testptp -d /dev/ptp0 -p 0
```

Configuration Example: 50 Hz Output

This signal format can be used to synchronize external equipment with a frequency higher than 1 Hz, if the ToD function is not used.

The pulse width is 50% if not specified with the -w option:

```
testptp -d /dev/ptp0 -p 20000000 -H 0
```

The pulse width is 1 ms:

```
testptp -d /dev/ptp0 -p 20000000 -H 0 -w 1000000
```

The pulse width is 1 ms and the phase is shifted +10 ns:

```
testptp -d /dev/ptp0 -p 20000000 -H 10 -w 1000000
```

The pulse width is 1 ms and the phase is shifted -10 ns:

```
testptp -d /dev/ptp0 -p 20000000 -H 19999990 -w 1000000
```

To disable the output:

```
testptp -d /dev/ptp0 -p 0
```

testptp Commands

```
# testptp --help
testptp: unrecognized option: -
usage: testptp [options]
 -c          query the ptp clock's capabilities
 -d name     device to open
 -e val      read 'val' external time stamp events
 -f val      adjust the ptp clock frequency by 'val' ppb
 -g          get the ptp clock time
 -h          prints this message
 -i val      index for event/trigger
 -k val      measure the time offset between system and phc clock
             for 'val' times (Maximum 25)
 -l          list the current pin configuration
 -L pin,val configure pin index 'pin' with function 'val'
             the channel index is taken from the '-i' option
             'val' specifies the auxiliary function:
             0 - none
             1 - external time stamp
             2 - periodic output
 -n val      shift the ptp clock time by 'val' nanoseconds
 -p val      enable output with a period of 'val' nanoseconds
 -H val      set output phase to 'val' nanoseconds (requires -p)
 -w val      set output pulse width to 'val' nanoseconds (requires -p)
 -P val      enable or disable (val=1|0) the system clock PPS
 -s          set the ptp clock time from the system time
 -S          set the system time from the ptp clock time
 -t val      shift the ptp clock time by 'val' seconds
 -T val      set the ptp clock time to 'val' seconds
 -z          test combinations of rising/falling external time stamp flags
```

9.5 ToD Output

The ToD output is a serial (ASCII or binary) based representation of the PTP time. It must always be accompanied by the 1PPS output, which indicates the precise phase of the time indicated by the serial ToD output. The serial ToD output is generated by the ToD daemon.

9.5.1 ToD Daemon

There is a software daemon provided (`todd`) which must be started to provide the ToD function.
The start options are listed by the help function:

```
# todd --h
Usage: todd [OPTION...]

  -a, --disable_announce_msg Disable G.8271 time announce message
  -d, --debug_level=number   Debug level: TOD_DBG_MSG=print all ..
                             (TOD_DBG_LVL_NUM - 1)=silent
  -i, --event_index=number   Specify the event to request, per default: 0
  -l, --log=path             Specify log file
  -m, --output_mode=mode     Specify output mode: GPZDA, GPZDG, G.8271
  -p, --ptp_device=device    Specify the ptp device to read
  -s, --socket_path=uds_path Specify UDS socket file for communication
  -u, --uart=device          Specify uart device
  -?, --help                 Give this help list
      --usage                Give a short usage message
  -V, --version              Print program version

Mandatory or optional arguments to long options are also mandatory or optional
for any corresponding short options.
```

Note: Preferably the ptp device is used (/dev/ptp0). It is also used for PTP time stamping.

Note: The output_mode accepts one of these strings: GPZDA, GPZDG, or G.8271.

The ToD daemon reads the time from the given extts source (usually `/dev/ptp2` of PON ToD).

When configured for binary G.8271 format, the ToD daemon needs more information than just the actual timestamp. For G.8271 event and announce messages, the ToD daemon reads this additional information from the `ptp4l` daemon.

9.5.2 Time of Day Configuration Options

Chapter 2.4.3.13 describe the adaptation to specific application requirements done in the UCI configuration.

9.6 Legacy ToD and 1PPS Output

This description is related to the 1PPS and ToD output directly from the PON IP hardware. This method is used up to SDK SYS_1.17.0. For a more flexible method provided from SYS_1.18.1 onwards see **Section 9.4**.

9.6.1 1PPS Output

The PON TC clock is synchronized to the OLT's clock through the mechanism provided by the ITU PON standards. The 1PPS signal is directly driven by the PON TC hardware.

The overall synchronization signal flow is shown in **Figure 142**.

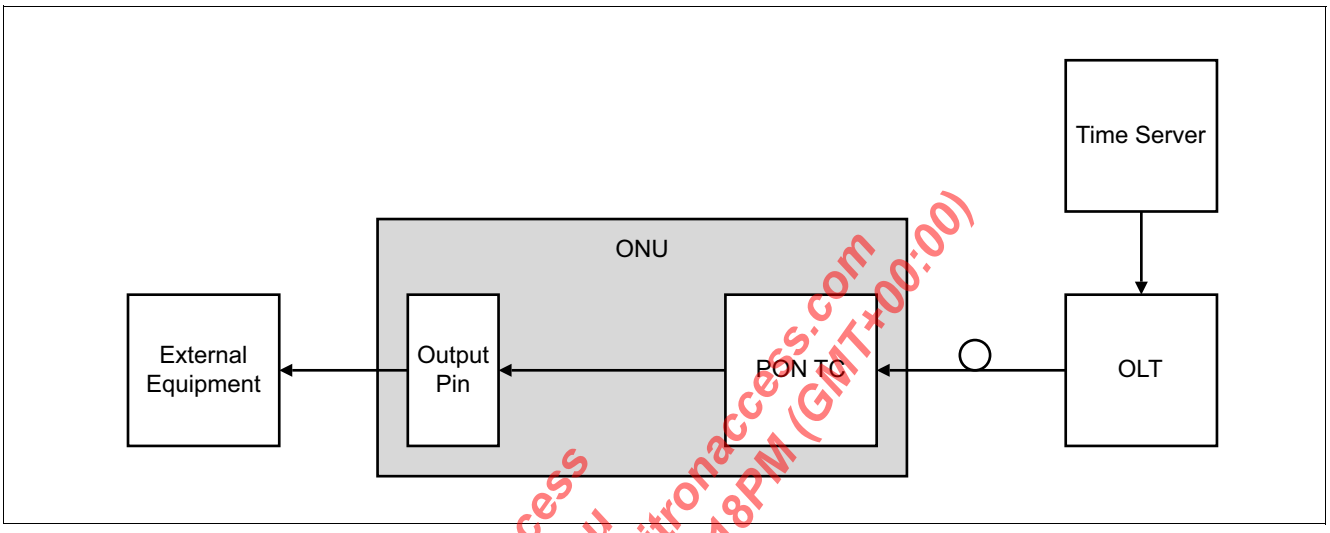


Figure 142 Clock and Time Synchronization Hardware Path – PON TC Driven

The 1PPS output is deactivated in the DTS by default. To make it usable, either the DTS must be modified before image compilation or, on an existing image, the command `pon pin_config_set 1 2` must be sent.

To further enable the 1PPS signal output, the PON library function `pon 1pps_event_enable` must be sent once before the first hardware synchronization is configured.

The output function of this pin must be configured in the Linux device tree (using `&pinctrl_gpc3`).

For example:

```
&ponmbox1 {
        status = "okay";
        pinctrl-0 = <&pinctrl_pon_ip &pinctrl_gpc3 &pinctrl_pon_ip_sfp_tx_dis>;
};

&pinctrl {
        pin_1pps_disabled: pin_1pps-disabled {
                intel,groups = "1pps_dis";
                intel,function = "gpio";
                intel,pins = <29>;
                intel,mux = <EQBR_MUX_0>;
        };
};
```

Figure 143 shows the signal output timing directly driven by the PON TC hardware.

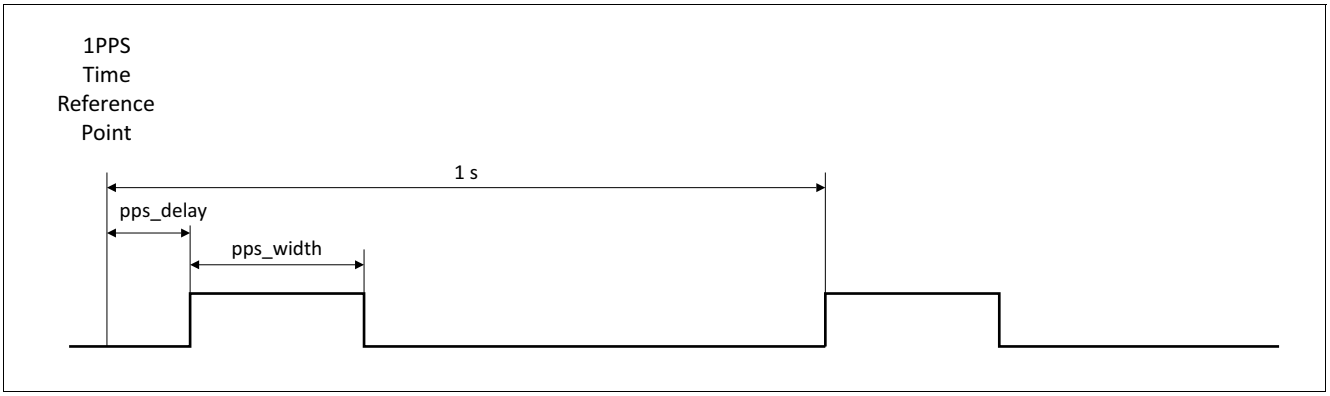


Figure 143 1PPS Timing – PON Driven

9.6.2 Time of Day Serial Output

The legacy ToD output format is configured through UCI, as described in Chapter 2.4.3.12.

10 Optical Diagnostics Monitoring

This chapter describes the digital diagnostics monitoring interface (DDMI) functions implementation for a PON ONU in an SFP form factor, based on PRX126 devices.

10.1 SFP Application Overview

The ONU devices integrated into an SFP form factor package provide management access to an external host controller through the two-wire (I^2C) interface bus. The SFP provides the slave on this bus, similar to a static EEPROM device. In addition to fixed status information, it provides dynamic data values to supervise the device temperature and optical receive power. **Figure 144** and **Figure 145** show the two main implementation options.

The first option provides full EEPROM space emulation, offering access to the address pages at addresses 0x50 and 0x51.

The second option places a physical EEPROM device into the SFP to serve the 0x50 address page while only the 0x51 page is emulated by the software. This option provides fast I^2C access after SFP insertion because there is no need to wait for the two-wire emulation software to be started.

To disable the emulation of address 0x50 by the software, an unused address must be configured during the slave firmware initialization.

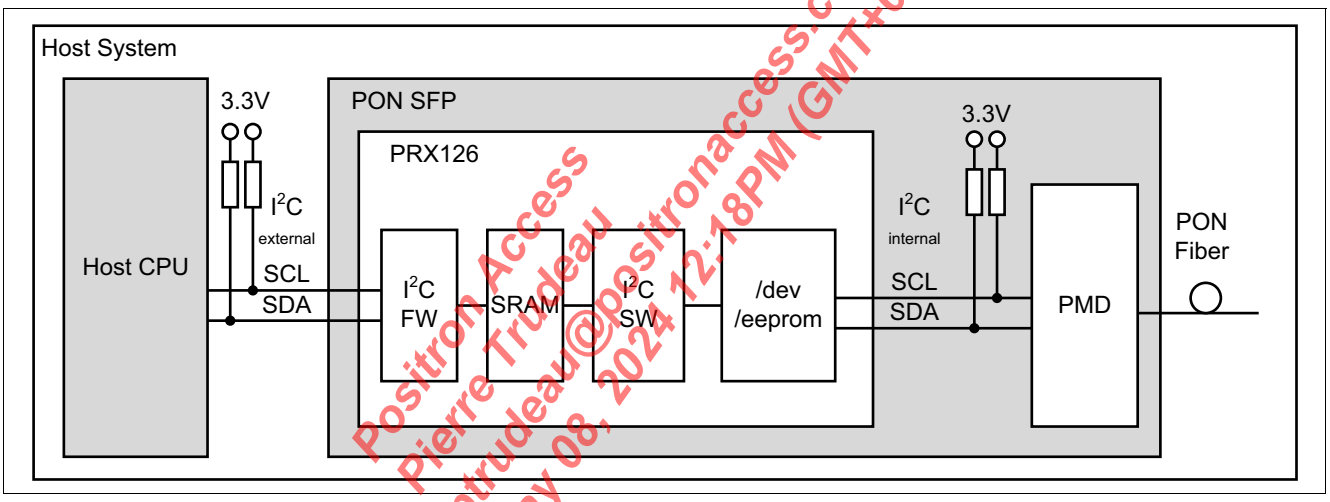


Figure 144 Dual Space EEPROM Emulation

See **Section 2.4.6.1.1** for more information.

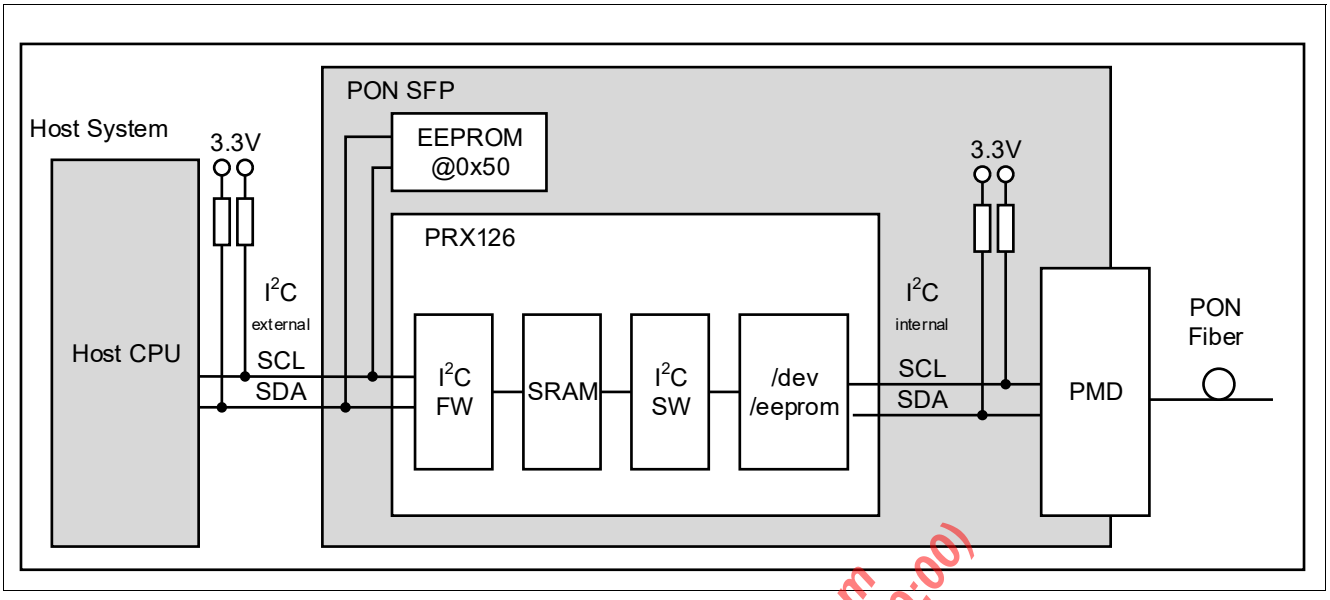


Figure 145 Single Space EEPROM Emulation

10.2 DDMI Functionality

The DDMI provides information about the optical interface status to an external host, into which the SFP ONU is plugged in. The access is granted through a two-wire serial interface as defined in SFF-8475 [67].

The data contents of the externally accessible memory space is internally held in an SRAM mapped to the data memory address space. The dedicated ioctl functions are available to access this memory space.

- `FIO_SFP_I2C_EEPROM_SET (sfp_eeprom_data)`
- `FIO_SFP_I2C_EEPROM_GET (sfp_eeprom_get)`
- `FIO_SFP_I2C_ADDR_SET (unsigned char)`
- `FIO_SFP_I2C_ADDR_GET (unsigned char)`
- `FIO_SFP_I2C_WP_SET (sfp_eeprom_range)`
- `FIO_SFP_I2C_WP_CLR (sfp_eeprom_range)`
- `FIO_SFP_I2C_WR_GET (sfp_eeprom_get)`
- `FIO_SFP_I2C_RD_GET (sfp_eeprom_get)`
- `FIO_SFP_I2C_EEPROM_CURRENT_SET`
- `FIO_SFP_I2C_WP_MASK_SET (sfp_eeprom_data)`
- `FIO_SFP_I2C_WP_MASK_GET (sfp_eeprom_get)`
- `FIO_SFP_I2C_EEPROM_MASKED_SET (sfp_eeprom_masked_data)`
- `FIO_SFP_I2C_PROCESSING_ENABLE (unsigned char)`

10.2.1 DDMI Status Information

The DDMI software updates some of the memory regularly to provide measured values related to the optical interface, such as temperature and received optical power. The implementation is based on a selected PMD module and requires adaptations when a different type is used. As the selected device is accessed through an internal two-wire bus in a common manner, these changes are supposedly minor.

10.2.2 DDMI Control Functions

There are certain control options provided for the external host and triggered by writing to selected memory locations. Write accesses to these locations are detected by the two-wire slave interface and signaled as an interrupt to the DDMI software.

The software handles these interrupts and performs the selected control functions.

These functions are supported:

- Read status information from address space 0xA0.
- Update of alarm and warning thresholds by the external host into address space 0xA2.
- Readout of measured values by the external host from address space 0xA2.
- Read and write to the status and control registers at address positions 0xA2-110_D and 0xA2-118_D.

10.2.3 Unsupported Functions

There are several options not supported by the implementation:

- Using parts of the address space as non-volatile memory.
- Support of vendor-specific functions.
- Serial speed modes other than 100 kHz or 400 kHz (standard mode).

10.2.4 Internal I²C Master Interface

The internal I²C master interface is under Linux driver control. The interface speed is selected to be 100 kHz or 400 kHz in the Linux device tree, according to the capabilities of the attached PMD device.

By default, the interface speed is set to 100 kHz.

10.2.5 DDMI Configuration

The application-specific settings are kept in a UCI configuration file.
The file is located at `/etc/config/sfp_eeprom`. This is an example configuration.

```
config 'sfp_eeprom' 'default'
       option 'vendor_name' 'Vendor Name ABCD'      # ASCII string, up to 16 characters
       option 'vendor_partno' '0123456789ABCDEF'    # ASCII string, up to 16 characters
       option 'vendor_rev' '0000'                   # ASCII string, up to 4 characters
       option 'vendor_oui' 'AC:9A:96'               # ASCII String, 3 hexadecimal
                                                    # numbers, colon separated
       option 'serial_no' '0123456789ABCDEF'        # ASCII string, up to 16 characters
       option 'datecode' '20190101'                 # ASCII string, up to 8 characters
       option 'addr_eeprom_0' '0x50'                # Hexadecimal value, 7 bit
       option 'addr_eeprom_1' '0x51'                # Hexadecimal value, 7 bit

config 'sfp_eeprom' 'factory_bridge'
       option 'addr_eeprom_0' '0x4E'                # Hexadecimal value, 7 bit
       option 'addr_eeprom_1' '0x4F'                # Hexadecimal value, 7 bit
       option 'enable' '<1|0>'                      # 0 (disabled) or 1 (enabled)
```

When the configuration is missing for one of these values, a default is used by the start-up script given by `/ugw_sw/feeds/ugw/pon/gpon-sfp-eeprom/files/sfp_eeprom.sh`.
The default values are defined in `/ugw_sw/feeds/ugw/pon/gpon-sfp-eeprom/Makefile` as:

```
DEFAULT_VENDOR_NAME="MaxLinear"
DEFAULT_VENDOR_PARTNO="0"
DEFAULT_VENDOR_REV="0001"
DEFAULT_VENDOR_OUI="00:50:F1"
DEFAULT_SERIAL_NO="no serial number"
DEFAULT_DATECODE="2021xxxx"
DEFAULT_DEVADDR0=0x50
DEFAULT_DEVADDR1=0x51
```

10.2.6 DDMI Software Startup

The DDMI software daemon is started through the start-up script `sfp_eeprom.sh`. This script reads the UCI configuration and starts the process.

10.2.7 Sub-Page Handling

The memory space 0x51/0xA2 allows the handling of multiple sub-pages, mapped to the higher 128 byte of the 256-byte address map. The address 0x1F holds the sub-page index used for subsequent accesses to this upper memory range.

Up to 5 sub-pages are supported, the sub-page index mapping to physical memory addresses must be configured in a table. There is a 256 x 1 byte table addressed by the logical sub-page index (0 to 255) which holds the value of the physical sub-page memory index (0 to 4). Any unsupported sub-pages are mapped to 0 (the default sub-page).

10.3 EEPROM Emulation

The external I²C interface as seen from the host system has no direct connection to the I²C bus of the optical interface. The software handles the read and write accesses performed by the external host into which the SFP is plugged in.

To allow fast access without clock stretching, which some host systems do not support, the interface subsystem provides a remotely accessible SRAM address range of 2 * 256 byte[1)]. This virtual EEPROM memory is externally accessed at two configurable I²C base addresses. The default values are 0xA0/0x50 and 0xA2/0x51.

The base addresses are defined in the UCI configuration.

Two memory spaces are referenced by these default base addresses.

The software checks the contents of the external EEPROM at the optical interface (PMD) regularly and sends changed values to the I²C slave module. The software does not check all byte locations, as most of them are static. Only the address ranges related to the optical interface receive and transmit parameters, temperature, and loss of signal are checked.

Each time the software detects a value change by the external host, it changes the related data values in the EEPROM of the PMD.

10.3.1 Address Definition

The configuration is held in the configuration file `/etc/config/sfp_eeprom`:

- `option 'pmd_0' '0x50'`
- `option 'pmd_1' '0x51'`

These variables are either set to a valid address (typically 0x50 and 0x51) when the emulation must be active or to an invalid address (0x80) when the emulation must be disabled for one or both address ranges and let an external EEPROM device answer to the valid address.

10.3.2 Initial Configuration

There is an option to modify the default settings within a start-up script called after booting.

The file name of the script is `/etc/init.d/sfp_eeprom.sh`.

10.3.3 Unsupported Features

These functions of a real physical EEPROM are not supported:

- Device identifier
  This 3-byte value read by the host through a special access method is not implemented.
- I²C general call function. This is not required because two I²C slave addresses are supported simultaneously.
- High-speed modes
  The clock frequencies on SCL above 400 kHz are not supported. This is a system limitation of the implemented firmware solution.
- Permanent storage of values written by the external host.

10.3.4 Memory Space Definition

This section defines the memory spaces exposed to the external I²C master device and how the individual values must be handled. The usage of the memory locations is based on the definitions by SFF-8472 Rev.12.2 and shown in **Table 81** and **Table 82**.

Figure 146 provides a visual overview of the memory ranges.

1) Including indirect sub-page access.

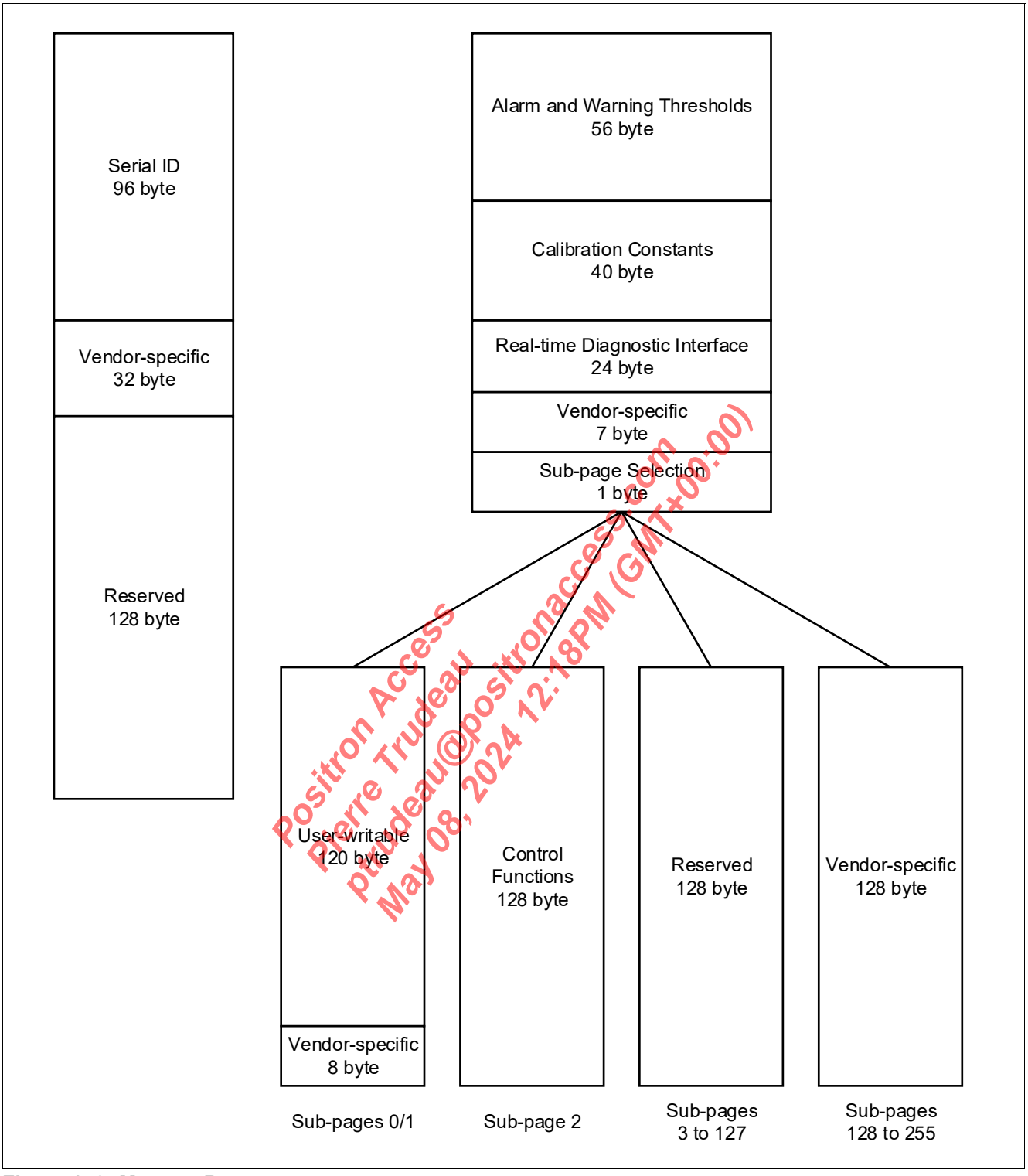


Figure 146 Memory Ranges

Table 81 Memory Space 0xA0

Address	Number of Bytes	R/W by Host	Name	Function	Software Handling[1)]
0	1	R	Identifier	Transceiver type description	Fixed value 0x03 (SFP or SFP+)
1	1	R	Extended Identifier	Transceiver type description	Fixed value 0x04 (two-wire interface ID)
2	1	R	Connector	Connector type code	Fixed value 0x01 (SC connector)
3-10	8	R	Transceiver	Transceiver compliance code bit map to indicate electrical and/or optical compatibility	Fixed value 0x00, to be updated by the OEM. ITU PON codes are not yet defined in SFF-8472.
11	1	R	Encoding	Describes the serial encoding	Fixed value 0x03 (NRZ)
12	1	R	Bit Rate	Nominal bit rate in units of 100 Mbit/s	Variable value, depending on the active PON operation mode. • G-PON: 25 • XG-PON: 100 • XGS-PON: 100 • NG-PON2: 100
13	1	R	Rate Identifier	Additional identification for special applications	Fixed value 0x00
14	1	R	Length SM km	Maximum single mode link length supported by the transceiver (x 1000 m)	Fixed value, in units of km, depending on the implemented optical interface. • 20 km: 20 • 40 km: 40 • 50 km: 50
15	1	R	Length SM hm	Maximum single mode link length supported by the transceiver (x 100 m)	Fixed value, in units of 100 m, depending on the implemented optical interface. • 20 km: 200 • > 25.4 km: 255
16	1	R	Length OM2	Maximum link length on 50 µm fiber (x 10 m)	Fixed value, in units of 10 m, depending on the implemented optical interface. G-PON technologies always support a fiber length exceeding this limit. • > 2.54 km: 255
17	1	R	Length OM1	Maximum link length on 62.5 µm fiber (x 10 m)	Fixed value, in units of 10 m, depending on the implemented optical interface. G-PON technologies always support a fiber length exceeding this limit. • > 2.54 km: 255
18	1	R	Length Copper	Minimum link length on copper cable (x 1 m)	Fixed value of 0x00, copper cables are not supported.

Table 81 Memory Space 0xA0 (cont'd)

Address	Number of Bytes	R/W by Host	Name	Function	Software Handling[1)]
19	1	R	Length OM3	Maximum link length on 50 µm fiber (x 10 m)	Fixed value, in units of 10 m, depending on the implemented optical interface. G-PON technologies always support a fiber length exceeding this limit. • > 2.54 km: 255
20-35	16	R	Vendor Name	Vendor name string (ASCII, left aligned, padded with 0x20)	Fixed value, defined by the OEM. ASCII string, left aligned (byte #20), padded with 0x20 to the right (byte #35).
36	1	R	Transceiver	Transceiver compatibility code	Fixed value, no coding specified for PON technologies, set to 0x00.
37-39	3	R	Vendor OUI	IEEE company identifier of the SFP vendor	Fixed value, 3-byte identifier set by the OEM, left aligned (leftmost byte at byte #37).
40-55	16	R	Vendor Part Number	SFP type identifier	Fixed value, defined by the OEM. ASCII string, left aligned (byte #40), padded with 0x20 to the right (byte #55).
56-59	4	R	Vendor Revision	SFP revision	Fixed value, defined by the OEM. ASCII string, left aligned (byte #56), padded with 0x20 to the right (byte #59).
60-61	2	R	Laser Wave Length	Nominal transmitter output wave length at room temperature [nm]	Fixed 16-bit integer value, MSB in byte #60, LSB in byte #61. Example: 0x04F6 = 1270 nm
62	1	R	Unused	No function	Fixed value, set to 0x00.
63	1	R	CC_BASE	Checksum over bytes 0 to 62 (including)	Recalculate each time one of the covered values changes.
64-65	2	R	Options	Bit map to indicate functional options offered by the device.	Fixed value per device type. This indicates the presence and polarity of the Signal Detect or Rx LOS pin and as well the presence of `TX_DISABLE` and `TX_FAULT`. LOS mode: Set to 0x001C SD mode: Set to 0x001A
66	1	R	BR max	Maximum bit rate [%]	Fixed value, upper limit given in % above nominal bit rate, set to 0x00 (unspecified).

Table 81 Memory Space 0xA0 (cont'd)

Address	Number of Bytes	R/W by Host	Name	Function	Software Handling[1)]
67	1	R	BR min	Minimum bit rate [%]	Fixed value, lower limit given in % above nominal bit rate, set to 0x00 (unspecified).
68-83	16	R	Vendor SN	Vendor serial number	Fixed value per device, vendor-specific format (ASCII, left justified, right padded with 0x20)
84-91	8	R	Date Code	Vendor production data code	Fixed value per device, set by the OEM at production time. Format: YYMMDD Year = 2000 + YY MM: 1 to 12 DD: 1 to 31
92	1	R	DDMI Type	Diagnostic monitoring type	Fixed value per device type, bit map to indicate the DDMI implementation. Set to 0x68 (internally calibrated, address range 0xA2 is available, average power value, addressing mode direct)
93	1	R	Enhanced Options	Option announcement for additional functionality	Fixed value per device type, defined by the OEM. Initialize to 0x00.
94	1	R	Compliance Code	SFF-8472 compliance information	Fixed value, set to 0x05.
95	1	R	CC_EXT	Checksum over bytes 64 to 94 (including)	Recalculate each time one of the covered values changes.
96-127	32	R/W	Vendor specific	Vendor-specific usage	This is a virtual EEPROM space for the external host controller. Initialize to values read from an initialization file. Each time the external host updates one of these bytes, the initialization file must be updated.
128-255	128	R	Reserved	Reserved for SFF-8079	Fixed value, set to 0x00 upon initialization.

1) See **Section 10.3.4.1** for more details.

Table 82 Memory Space 0xA2

Address	Number of Bytes	R/W by Host	Name	Function	Software Handling
Alarm and Warning Thresholds					
0-1	2	R/W	High Alarm	Temperature thresholds	Reconfigurable values. The software reads thresholds and compares them against measured values. Set warning/alarm flags, when thresholds are exceeded.
2-3	2	R/W	Low Alarm		
4-5	2	R/W	High Warning		
6-7	2	R/W	Low Warning		
8-9	2	R/W	High Alarm	Voltage thresholds	
10-11	2	R/W	Low Alarm		
12-13	2	R/W	High Warning		
14-15	2	R/W	Low Warning		
16-17	2	R/W	High Alarm	Bias current thresholds	
18-19	2	R/W	Low Alarm		
20-21	2	R/W	High Warning		
22-23	2	R/W	Low Warning		
24-25	2	R/W	High Alarm	Transmit power thresholds	
26-27	2	R/W	Low Alarm		
28-29	2	R/W	High Warning		
30-31	2	R/W	Low Warning		
32-33	2	R/W	High Alarm	Receive power thresholds	
34-35	2	R/W	Low Alarm		
36-37	2	R/W	High Warning		
38-39	2	R/W	Low Warning		
40-55	2	R	Reserved	None	Fixed value, initialize to 0x0000.
56-91	2	R	Calibration Data	Used for external calibration	Fixed value, initialize to 0x0000.
92-94	2	R	Reserved	None	Fixed value, initialize to 0x0000.
95	1	R	Checksum	Checksum over bytes 0 to 94 (including)	Recalculate each time one of the covered values changes.
Measured Values					
96	1	R	Temp MSB	Internally measured module temperature	Report the temperature value derived from the internal temperature sensor.
97	1	R	Temp LSB		
98	1	R	Vcc MSB	Internally measured supply voltage	Report the internal SFP voltage value.
99	1	R	Vcc LSB		
100	1	R	Tx Bias MSB	Internally measured bias current	Report the transmit laser bias current value.
101	1	R	Tx Bias LSB		
102	1	R	Tx Power MSB	Measured transmit output power	Report the optical transmit power value.
103	1	R	Tx Power LSB		

Table 82 Memory Space 0xA2 (cont'd)

Address	Number of Bytes	R/W by Host	Name	Function	Software Handling
104	1	R	Rx Power MSB	Measured receive input power	Report the optical receive power value.
105	1	R	Rx Power LSB		
106-109	4	R	Reserved	None	Fixed value, initialize each byte to 0x00.
Status and Control					
110	1	R/W	Status and Control	Miscellaneous status and control functions	Report status and react on control bit changes. Bit 7_D **RD** Tx disable pin status Bit 6_D **RD/WR** Soft Tx_disable enable Bit 5_D **RD** RS(1) pin status Bit 4_D **RD** RS(0) pin status Bit 3_D **RD/WR** Soft rate_select enable Bit 2_D **RD** TX_FAULT pin status Bit 1_D **RD** RX_LOS pin status Bit 0_D **RD** DATA_READY_BAR pin status
111	1	R	Reserved	None	Fixed value, initialize to 0x00.
112	1	R	Alarm Flags	Report alarm flags for Temperature, Vcc, Tx bias, Tx power	Report alarm flags, when the alarm threshold is exceeded.
113	1	R	Alarm Flags	Report alarm flags for Rx optical power	
114	1	R	Reserved	None	Fixed value, initialize to 0x00.
115	1	R	Reserved	None	Fixed value, initialize to 0x00.
116	1	R	Warning Flags	Report warning flags for Temperature, Vcc, Tx bias, Tx power	Report warning flags, when the alarm threshold is exceeded.
117	1	R	Warning Flags	Report warning flags for Rx optical power	
118	1	R/W	Extended Status and Control	Miscellaneous status and control functions	Select and report the power dissipation level. Bit 7_D **RD** Reserved Bit 6_D **RD** Reserved Bit 5_D **RD** Reserved Bit 4_D **RD** Reserved Bit 3_D **RD/WR** Soft RS(1) enable Bit 2_D **RD** Reserved Bit 1_D **RD** Power level status Bit 0_D **RD/WR** Power level control
119	1	R	Reserved	None	Fixed value, initialize to 0x00.
Vendor Space					
120-126	8	R/W	Vendor Memory	Vendor-specific memory space	Fixed values, to be defined by the OEM. When not, initialize to 0x00.

Table 82 Memory Space 0xA2 (cont'd)

Address	Number of Bytes	R/W by Host	Name	Function	Software Handling
127	1	R/W	Page	Page selection for the upper address range.	Functional implementation for page a limited number of pages.
Page Selection = 0x00 or 0x01					
128-247	120	R/W	User EEPROM	Readable and writable EEPROM space, accessible by the external host.	This is a virtual EEPROM space for the external host controller. Nonvolatile storage availability depends on the connected PMD device.
248-255	8	R/W	Vendor control	Vendor-specific control space	Fixed or variable values, to be defined by the PMD vendor. Initialized to fixed values of 0.
Page Selection = 0x02 to 0xFF					
128-255	128	R	Control Functions	Vendor-specific control function enhancements.	Fixed or variable values, to be defined by the PMD vendor. Initialized to fixed values of 0.

10.3.4.1 Default Value Definition

The read-only default values to be shown in the 0xA0 address range are defined in the header file `/gpon_sfp_eeprom/src/sfp_i2c_eeprom_defaults.h`.

Application-specific adaptation is done in this file.

When the initialization script `/etc/init.d/sfp_eeprom.sh` contains this code in the `default_sfp_init()` section, the default values are used:

```
# reset to default values
sfp_i2c -d yes
```

Another option to modify values is to use the `sfp_i2C()` function in the initialization file. For example:

```
sfp_i2c -1 -l 1 -i 3 -w 0xZZ
```

10.3.5 Read and Write Access Definition

The driver functions `FIO_SFP_I2C_WP_SET` and `FIO_SFP_I2C_WP_MASK_SET` are used to specify a single address location or address ranges to be write-protected (read-only by the external host).

All address locations are readable by the external host.

All address locations not explicitly configured to be write-protected are writable.

10.3.6 Numeric Representation of Measured Values

The reporting of measured values uses the coding as shown hereafter.

The values are copied from the optical module at regular times (1 s).

10.3.6.1 Transceiver Temperature

Represented as a 16-bit signed two complement value in increments of 1/256 Celsius degrees, yielding a total range of -128 °C to +128 °C.

The temperature in degrees Celsius is given by the signed two complement value with LSB equal to 1/256 °C.

10.3.6.2 Transceiver Supply Voltage

Represented as a 16-bit unsigned integer with the voltage defined as the full 16-bit value (0 to 65535) and with LSB equal to 100 µV, yielding a total range of 0 to +6.55 V.

10.3.6.3 Laser Bias Current

Represented as a 16-bit unsigned integer with the current defined as the full 16-bit value (0 to 65535) and with LSB equal to 2 µA, yielding a total range of 0 to 131 mA.

10.3.6.4 Transmit Output Power

Represented as a 16-bit unsigned integer with the power defined as the full 16-bit value (0 to 65535) and with LSB equal to 0.1 µW, yielding a total range of 0 to 6.5535 mW (~ -40 to +8.2 dBm).

10.3.6.5 Received Optical Power

Represented as a 16-bit unsigned integer with the power defined as the full 16-bit value (0 to 65535) and with LSB equal to 0.1 µW, yielding a total range of 0 to 6.5535 mW (~ -40 to +8.2 dBm).

10.3.7 Factory Mode

For calibration and initialization purposes during production, a bridge operation mode is implemented. In this mode, a direct access to the PMD device is granted to the external host for initial configuration.

The bridge mode is selected by the UCI option `'factory_bridge' 'enable' '1'`.

11 Home Gateway Units

The implementation of a Home Gateway Unit (HGU) implies some specific differences to the SFU and SFP implementation that have been described before.

This chapter describes the basic differences between an HGU compared to an SFU based on the UGW software implementation. For specifics of RDK-B or prplOS see **Chapter 12** and **Chapter 13**, respectively.

If not explicitly mentioned in these chapter, the description given here applies as well.

11.1 Data Forwarding Operation Modes

This section describes the different HGU data forwarding modes.

11.1.1 Routing Mode

This is the regular residential gateway operating mode. User data is routed between the UNI interfaces and the VEIP interface to the PON data layer.

11.1.2 Bridged Mode

The UNI-side interfaces are modeled as PPTP Ethernet UNI managed entities.

This operation mode is currently not supported.

11.1.3 Mixed Bridged and Routed Mode

Both VEIP and PPTP UNI managed entities can be used at the UNI side (see **Figure 147** and **Figure 148**).

This operation mode is currently not supported.

11.1.4 Bypass Mode for Interoperability Testing

Some interoperability tests, such as BBF TP-247, require a bridged traffic forwarding without NAT or other routing functionality, although the OMCI model uses a VEIP and no PPTP Ethernet UNI managed entities. There is a dedicated “BBF Mode 1” that can be selected in this way:

```
echo VANI0 VUNI0_0 eth0_1 enable > /sys/kernel/debug/ppa/hal/pp/bbf247_hgu_model
```


11.1.5 OMCI HGU Data Model

The basic OMCI data model for an HGU implementation is shown in **Figure 147** and **Figure 148** for the downstream and upstream data direction, respectively.

Multiple VEIP and/or UNI interfaces can be connected to a single bridge or multiple bridges. Multiple GEM ports and T-CONTs can be instantiated to handle the upstream QoS for different types of service. In downstream direction dedicated multicast GEM ports can be created to handle the IGMP/MLD-controlled multicast traffic.

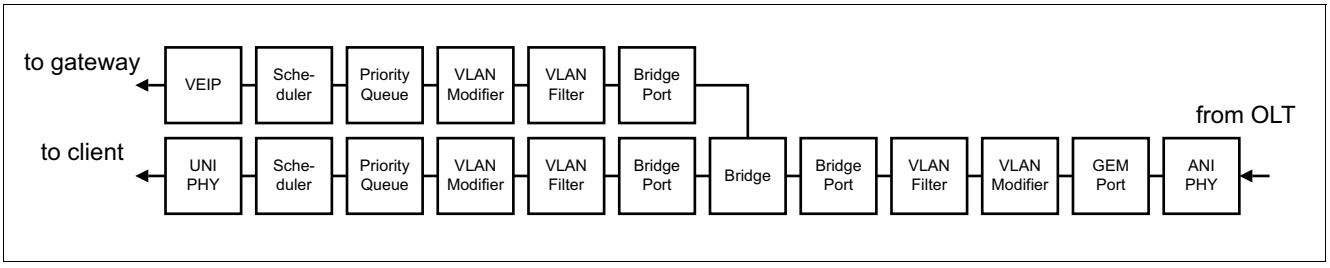


Figure 147 Basic Downstream HGU OMCI Model

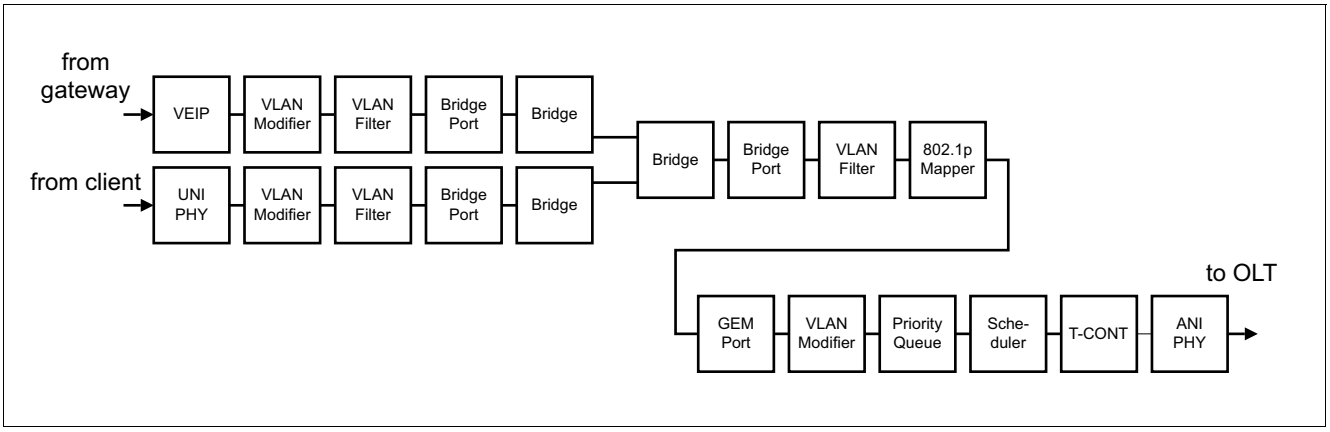


Figure 148 Basic Upstream HGU OMCI Model

11.2 OMCI Configuration Options

The OMCI models the HGU as a multi-managed ONU device. The demarcation between the management domains is the Virtual Ethernet Interface Point (VEIP). The local LAN interfaces are not under OMCI control but managed by the gateway management system.

This is reflected in the MIB initialization file where the PPTP Ethernet UNI is replaced by the VEIP managed entity. For example:

```
# Virtual Ethernet Interface Point
329 0x0101 0 0 \0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0 0 0xffff

## VEIP Cardholder
5 0x0101 48 48 0 "                    " "                    " 0 0 0 0

## VEIP Circuit Pack
6 0x0101 48 1 "        " "             " VEND 0 1 0 "                    " 0 0 0 0 0
```

Due to restrictions of some OLT types that are unable to handle the VEIP managed entity, it is also possible to instantiate a PPTP Ethernet UNI instead of the VEIP. Such as in this example:

```
## PPTP Ethernet UNI
11 0x0101 0 0 0 0x00 1 1 0 1534 0 0xffff 0 0 0 0 0

## UNI Cardholder
5 0x0101 49 49 0 "                    " "                    " 0 0 0 0

## UNI Circuit Pack
6 0x0101 49 1 "        " "             " VEND 0 1 0 "                    " 0 0 0 0 0
```

As long as there is no need to interoperate with a specific, limited OLT type, the VEIP approach should be used.

11.3 Software Upgrade

On an HGU system the software upgrade can be triggered by multiple sources, which are for example

- OLT through OMCI
- ACS through CWMP
- End user through WebUI

To align these different applications, a software upgrade daemon has been implemented that is controlled through *ubus* commands **[38]**.

The software upgrade process supports three different handling methods:

- Toggle operation
  - Two image versions are stored and each of them can be selected to be used.
    This allows to fall back to an older software version if needed.
    This method is used for OMCI-controlled software upgrade.
- Backup operation
  - Before an update, the current image is copied from the primary partition to a backup partition.
    If the upgrade process fails, the backup image can be copied back and run again from the primary partition.
- Redundant operation
  - Two copies of the software image are held in the non-volatile memory.
    This allows to resume operation if one of the images is damaged due to storage issues.

An overview of the data flow is shown in **Figure 149**.

Table 83 lists the details of the *ubus* procedures and arguments.

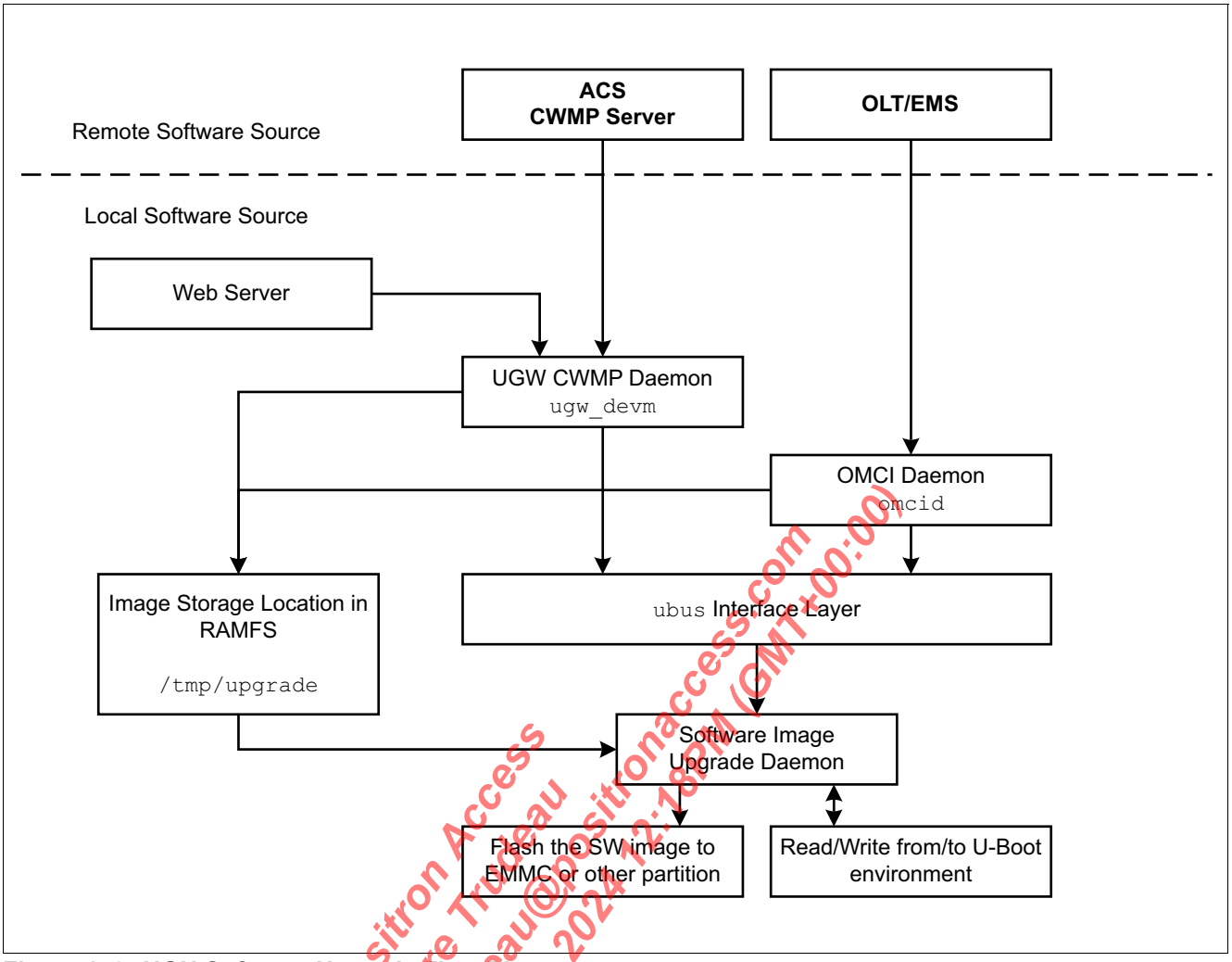


Figure 149 HGU Software Upgrade Flow Diagram

Table 83 Software Upgrade ubus Methods and Policies

Procedures	Arguments	Values	Usage
img_activate	bank	A or B	Activate one of the two images.
get_uboot_env	–	–	Returns a list of U-Boot environment variable values: • active_bank • commit_bank • img_type • img_versionA • img_versionB • img_validA • img_validB

Table 83 Software Upgrade ubus Methods and Policies (cont'd)

Procedures	Arguments	Values	Usage
get_uboot_param	String to select one of the U-Boot environment variables	• get_active_bank • get_commit_bank • get_img_versionA • get_img_versionB • get_img_type • get_img_validA • get_img_vallidB	Returns the value of a selected U-Boot environment variable.
set_uboot_env	String(s) to select one or more of the U-Boot environment variables	• set_commit_bank (string "A" or "B") • set_img_type (integer 0, 1, 2) • set_img_versionA/B (string up to 14 characters) • set_img_versionB (string up to 14 characters) • set_img_validA (boolean, 0 or 1) • set_img_validB (boolean, 0 or 1) *Note: set_img_versionA/B and set_img_validA/B are debug functions. The values are set automatically by the "write_img" function.*	Modify one or more of the U-Boot environment variables. The return values are defined as: • 0: Success • 1: Invalid parameter • 2: Operation failed The image types are defined as: • 0: Toggle mode • 1: Backup mode • 2: Redundant mode
write_img	bank	A or B	Trigger the copy operation of the file `/tmp/upgrade/firmware.img` to the selected bank of the flash memory. The return values are defined as: • 0: Success • 1: Image validation failure • 2: Write to flash failure
reboot	–	–	Reboot the system according to the configured U-Boot environment variables.

11.4 Network Interfaces

The network interface structure differs from the SFU application, because there are more network interfaces needed for an HGU. This is a example list:

```
root@ugwcpe:~# ip link
ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN mode DEFAULT group default qlen
1000
    link/ipip 0.0.0.0 brd 0.0.0.0
3: gre0@NONE: <NOARP> mtu 1476 qdisc noop state DOWN mode DEFAULT group default qlen
1000
    link/gre 0.0.0.0 brd 0.0.0.0
4: gretap0@NONE: <BROADCAST,MULTICAST> mtu 1462 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
5: erspan0@NONE: <BROADCAST,MULTICAST> mtu 1450 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
6: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN mode DEFAULT group default qlen
1000
    link/sit 0.0.0.0 brd 0.0.0.0
7: ip6tnl0@NONE: <NOARP> mtu 1452 qdisc noop state DOWN mode DEFAULT group default
qlen 1000
    link/tunnel6 :: brd ::
8: ip6gre0@NONE: <NOARP> mtu 1448 qdisc noop state DOWN mode DEFAULT group default
qlen 1000
    link/gre6 :: brd ::
9: vni0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN
mode DEFAULT group default qlen 1000
    link/ether 16:1a:4d:20:a7:20 brd ff:ff:ff:ff:ff:ff
10: ins0: <BROADCAST,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UNKNOWN mode DEFAULT group default qlen 1000
    link/void 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
11: dummy0: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN mode DEFAULT group
default qlen 1000
    link/ether 0e:55:9e:42:39:81 brd ff:ff:ff:ff:ff:ff
12: ifb0: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN mode DEFAULT group default
qlen 32
    link/ether b6:3e:f9:bf:8c:df brd ff:ff:ff:ff:ff:ff
13: ifb1: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN mode DEFAULT group default
qlen 32
    link/ether 66:0e:23:59:ad:91 brd ff:ff:ff:ff:ff:ff
14: bond0: <BROADCAST,MULTICAST,MASTER> mtu 1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/ether 6a:1e:74:31:83:d0 brd ff:ff:ff:ff:ff:ff
15: teql0: <NOARP> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 100
    link/void
16: intel_vpn: <> mtu 10034 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/netrom
```

```
17: eth0_1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master br-lan state
UP mode DEFAULT group default qlen 1000
    link/ether 02:9a:96:f0:9e:70 brd ff:ff:ff:ff:ff:ff
18: eth0_2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq master br-lan
state DOWN mode DEFAULT group default qlen 1000
    link/ether 02:9a:96:f0:9e:71 brd ff:ff:ff:ff:ff:ff
19: eth0_3: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq master br-lan
state DOWN mode DEFAULT group default qlen 1000
    link/ether 02:9a:96:f0:9e:72 brd ff:ff:ff:ff:ff:ff
20: eth0_4: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq master br-lan
state DOWN mode DEFAULT group default qlen 1000
    link/ether 02:9a:96:f0:9e:73 brd ff:ff:ff:ff:ff:ff
21: eth0_5: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq master br-lan
state DOWN mode DEFAULT group default qlen 1000
    link/ether 02:9a:96:f0:9e:74 brd ff:ff:ff:ff:ff:ff
22: VANI0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UNKNOWN mode
DEFAULT group default qlen 1000
    link/ether 02:9a:96:f0:9e:76 brd ff:ff:ff:ff:ff:ff
23: VUNI0_0: <BROADCAST,MULTICAST> mtu 1516 qdisc noop state DOWN mode DEFAULT group
default qlen 1000
    link/ether ac:9a:96:f0:9e:71 brd ff:ff:ff:ff:ff:ff
24: VUNI0_1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group
default qlen 1000
    link/ether 7e:d0:80:e1:d9:c8 brd ff:ff:ff:ff:ff:ff
25: VUNI0_us: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group
default qlen 1000
    link/ether ee:af:72:04:66:bf brd ff:ff:ff:ff:ff:ff
26: VUNI0_2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group
default qlen 1000
    link/ether 56:4b:84:eb:bb:9a brd ff:ff:ff:ff:ff:ff
27: rtlog0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group
default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
28: pon0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group
default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
29: mcdev0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group
default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
30: lpdev0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UNKNOWN mode
DEFAULT group default qlen 1000
    link/ether ca:83:e6:88:05:ab brd ff:ff:ff:ff:ff:ff
31: vpnnf_dev0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
32: br-lan: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode
DEFAULT group default qlen 1000
    link/ether 02:9a:96:f0:9e:70 brd ff:ff:ff:ff:ff:ff
33: tcont-omci@pon0: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop state DOWN
mode DEFAULT group default qlen 1000
    link/ether 82:bb:3b:e7:7b:88 brd ff:ff:ff:ff:ff:ff
```

```
34: gem-omci@pon0: <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP,M-DOWN> mtu 2030 qdisc
pfifo_fast state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether 02:50:f1:00:00:01 brd ff:ff:ff:ff:ff:ff
```

From a PON perspective, these network interfaces are of interest:

- `VANI0`
  – This is the WAN interface to the PON. All data traffic to the WAN crosses this interface.
- `VUNI0_0`
  – This interface is used for downstream unicast traffic.
- `VUNI0_1`
  – This interface is used for downstream multicast traffic.
- `VUNI0_2`
  – This interface is used for downstream broadcast traffic.
- `tcont-omci`
  – This is the T-CONT that is used for OMCI management packets (OMCC).
- `gem-omci`
  – This is the GEM port that is used for OMCI management packets (OMCC).
  OMCI packets received from or addressed to the OLT are received from or sent to this interface.

See **Chapter 3.1** for more details on network devices.

11.5 System Initialization

The system initialization uses the UCI configuration files located at `/etc/config`.

In contrast to the SFU applications based on the PRX device family, the HGU applications based on the URX devices do not store the UCI files persistently but re-create them upon system start by the Configuration Storage Daemon (CSD) from a Data Base (DB). This provides additional security related to a possible tampering of UCI files.

To modify UCI values persistently, the CSD must be use through `ubus` messages..

For more detail refer to the UGW documentation **[18]**.

12 RDK-B System Differences

The Reference Design Kit for Broadband (RDK-B) [35] is an open source Linux-based framework that has been developed to support broadband gateway implementations.

The implementation of a Home Gateway Unit (HGU) based on RDK-B differs in some aspects from the UGW implementation. This chapter describes the major differences. Other ONU types (SFU, MDU, etc.) are not supported for RDK-B.

12.1 Build System

RDK-B uses the Yocto project as a base and the OpenEmbedded build system [36]. To access the git resources it needs a personal registration with MaxLinear Design Center and RDK-B Central.

Refer to the RDK-B and Yocto documentation for more detail.

12.2 System Initialization

Refer to the RDK-B documentation.

12.2.1 Device Tree

See Chapter 2.4.2.

12.2.2 Boot Loader

As in OpenWrt, the U-Boot used as the system boot loader.

12.3 RDK-B PON Hardware Abstraction Layer

The RDK-B HAL is a software interface that allows access to PON status information, independent of the underlying hardware and software model. It is based on a JSON client/server architecture and the data model is defined by a JSON file.

The 10G PON Chipset System Package provides the JSON server and the higher layer PON management stack must provide the JSON client to access the data.

12.4 Daemon Handling

`systemd` is used to start, stop, and restart daemons. The `/etc/init.d` files are replaced by `systemd` unit files. Logfiles are stored in `journald` (which is part of `systemd`). A separate log is stored for each daemon.

13 prplOS System Differences

The prpl Foundation maintains an open-source framework on GitHub [30] that supports the implementation of broadband gateway solutions.

The implementation of a Home Gateway Unit (HGU) based on prplOS differs in some aspects from the UGW implementation. Other ONU types (SFU, MDU, etc.) are not supported for prplOS.

General information for prplOS can be found at the prplOS Wiki page [31].

13.1 Build System

Both the prplOS as well as the UGW system are based on OpenWrt.

The differences are described in [34].

These are the steps to build an image for URX851:

```
git clone https://gitlab.com/prpl-foundation/prplos/prplos
cd prplos
git checkout UPDK_8.9.20
./scripts/gen_config.py mxl_x86 webui prpl
make -j8
```

Note: Access permission to git.maxlinear.com is required.

This will generate the image

```
./bin/targets/intel_x86/lgm/openwrt-intel_x86-lgm-LGM_PRPL-
lgp_b0_pon_fullimage.img
```

13.1.1 Image Loading

The image can be loaded manually by using these commands:

```
# mmc erase ${overlay_container_a_block_start} ${overlay_container_a_block_size}
# setenv fullimage openwrt-intel_x86-lgm-LGM_PRPL-lgp_b0_fullimage.img
# saveenv
# run update_fullimage
# run bootcmd
```


13.2 System Initialization

Refer to the prplOS documentation.

13.2.1 Device Tree

See Chapter 2.4.2.

13.2.2 Boot Loader

As in OpenWrt, the U-Boot used as the system boot loader.

13.2.3 Configuration Database

As in OpenWrt, the UCI files are used as the system configuration data. See Chapter 2.4.3 for reference.

13.3 prplOS PON Hardware Abstraction Layer

The prplOS PON HAL is a software interface that allows access to PON status information, independent of the underlying hardware and software model. It is based on a `ubus` inter-process communication that uses a PON data model. This data model is part of the 10G PON Chipset System Package and can be found in **[33]**.

The `ubus` protocol is an integral part of the prplOS system and supported by a dedicated library (`libubus`).

13.4 Daemon Handling

The logs of the OMCI daemon (`omcid`) are stored into a logfile instead of `stdout`.

14 Debug Functions

This chapter describes the debug options provided for the implementation of 10G PON ONU systems. A more general description of UGW debugging is given in [22].

Note: Several debug commands use the `io` *function for direct register or memory access. For software security reasons, this function is by default* ***not*** *included in the software binary and must be enabled explicitly by the related build option. Such a software binary* ***must not*** *be used for productive software delivery. See Section 15.1.1 for details.*

14.1 Local Serial Access

There are two serial interface ports available for direct connection to a debug station. Both interfaces are configured to dedicated internal submodules as well as to the Linux command line.

Attention: In a secure image, the access to the serial interfaces is blocked by default.

These debug options are available for the two interfaces:

- UART0 (URXD0, UTXD0)
  - PON module interface #0
  - QoS module
  - GPHY module
- UART1 (URXD1, UTXD1)
  - PON module interface #1
  - Voice codec
  - Secure Boot processor (TEP)

14.1.1 Switch from UART0 to UART1

When the UART1 interface is physically not available, the physical output directed to UART1 is temporarily switched to the physical output of UART0. For example, this is used to test the ToD serial output.

This is an example to switch for 60 seconds:

```
# io -4 0x16180120 0x00d84016; sleep 60; io -4 0x16180120 0x00d80016
```

Note: The temporary switching is required because when UART1 is activated there is no manual option to switch back.

14.1.2 Disable the UART0 Access

The access to UART0 can be disabled by defining a U-Boot environment variable:

```
fw_setenv uart_select_preboot off
reboot
```

With the next reboot the UART interface is deactivated.

To re-enable the UART interface, connect to the Linux command line over an SSH connection and call

```
fw_setenv uart_select_preboot
reboot;exit
```

14.1.3 Enable ToD Output on UART0

To enable ToD serial output on UART0, define the U-Boot variable:

```
fw_setenv uart_select_preboot tod
reboot
```

This will disable the Linux command line access through UART0.

To re-enable the Linux command line on UART0, connect to the Linux command line over an SSH connection and call

```
fw_setenv uart_select_preboot
reboot;exit
```

14.2 Local and Remote Command Line Access

Using an Ethernet and IP connection, access to the Linux command line is possible through any of the local Ethernet interfaces or the OLT Ethernet interface over the PON fiber. Depending on the software configuration, telnet or SSH is used as the remote access protocol.

SSH is the preferred access method and a secure password must be set.

Attention: telnet must never be included in productive software due to its inherent security risks.

14.3 Command Line Functions

Several of the software libraries provide a command line option. This is used to enter library functions from the command line for testing and debug purposes.

14.3.1 Hardware, Software, and Firmware Versions

These commands are provided to check the system software and hardware versions.

Note: These are example outputs, the version number differ depending on the software release.

SDK Release

```
# cat /etc/pon.ver
1.22.0
```

U-Boot Version

```
# fw_printenv ver
ver=U-Boot 2016.07-MXL-v-3.1.273 (…)
```

Hardware Type Information

```
# cat /proc/cpuinfo | grep system
system type             : PRX300 rev 1.1

# cat /proc/cpuinfo | grep machine
machine                 : PRX321-SFU-QSPI-PON
```

Note: This depends on the hardware on which the command is called.

```
# cat /proc/cpuinfo | grep cpu
cpu model               : MIPS interAptiv (multi) V2.0

# pm_util -a | grep "CPU frequency"
 current CPU frequency: [info 800 MHz] [scale 800 MHz] [max 800 MHz]
 current CPU frequency: [info 800 MHz] [scale 800 MHz] [max 800 MHz]
 current CPU frequency: [info 800 MHz] [scale 800 MHz] [max 800 MHz]
 current CPU frequency: [info 800 MHz] [scale 800 MHz] [max 800 MHz]
```

Linux Kernel Version

```
# uname -r
4.9.337+
```

Linux Version and Compile Information

```
# cat /proc/version
Linux version 4.9.337+ (hostname) (gcc version 11.2.0 (OpenWrt GCC 11.2.0
v22.03.3_maxlinear) ) …
```

Other basic system information are found at `/proc`.

Linux Boot Logfile

Use the `logread` command to print the Linux boot log.

OpenWrt/LEDE Version

```
# cat /etc/openwrt_release
DISTRIB_ID='LEDE'
DISTRIB_RELEASE='1.22.0'
DISTRIB_REVISION='v22.03.3_maxlinear'
DISTRIB_TARGET='intel_mips/prx300'
DISTRIB_ARCH='mips_24kc_nomips16'
DISTRIB_DESCRIPTION='LEDE 1.22.0 r20028-43d71ad93e'
DISTRIB_TAINTS='no-all busybox'
```

Operating System Version

```
# cat /etc/os-release
NAME="LEDE"
VERSION="1.22.0"
ID="lede"
ID_LIKE="lede openwrt"
PRETTY_NAME="LEDE 1.22.0"
VERSION_ID="1.22.0"
HOME_URL="https://openwrt.org/"
BUG_URL="https://bugs.openwrt.org/"
SUPPORT_URL="https://forum.openwrt.org/"
BUILD_ID="v22.03.3_maxlinear"
OPENWRT_BOARD="intel_mips/prx300"
OPENWRT_ARCH="mips_24kc_nomips16"
OPENWRT_TAINTS="no-all busybox"
OPENWRT_DEVICE_MANUFACTURER="OpenWrt"
OPENWRT_DEVICE_MANUFACTURER_URL="https://openwrt.org/"
OPENWRT_DEVICE_PRODUCT="MaxLinear PRX321 / PRX126 System"
OPENWRT_DEVICE_REVISION="v0"
OPENWRT_RELEASE="LEDE 1.22.0 r20028-43d71ad93e"
```


Software Module Versions

Call this shell script to show the image version and a list of software module versions[1]:

```
# show_version.sh
------------------------------------------------------------------------
 MaxLinear PRX321 / PRX126 System, Image Revision 1.22.0
------------------------------------------------------------------------
PON subsystem version: 1.22.0
CLI Library, Version 2.9.0.
KPI2UDP driver, version 3.6.0.0
MaxLinear PON Adapter, version 1.17.3
MaxLinear PON Image Library, Version 2.4.0
MaxLinear PON Mailbox driver, version 1.24.0
MaxLinear PON Multicast driver, version 1.12.1
MaxLinear PON Net Lib, version 1.22.1
MaxLinear PON SFP I2C driver, version 1.2.1
MaxLinear PON library CLI, version 1.26.1.0
MaxLinear PON library daemon, version 1.26.1.0
```

1) On URX-based designs the command is “version.sh”.

```
MaxLinear PON library, Version 1.26.1.0
MaxLinear PON network driver, version 1.24.0
MaxLinear PON top, version 1.11.0
MaxLinear SFP EEPROM Daemon, version 3.6.0
OMCI Simulate, version 8.19.1
OMCI daemon, version 8.19.1
SyncE tool, version 0.1.5
TAPI, Version 4.26.0.0
ToD daemon, version 1.4.1
VMMC device driver, version 1.35.0.0
```

PON IP Hardware and Firmware Version

```
# pon version_get
errorcode=0 hw_version=7 sw_version=72193 fw_version_major=3 fw_version_minor=25
fw_version_standard=0 fw_version_platform=3 fw_version_patch=0 fw_version_is_test=0
fw_timestamp=1704788523
```

- `hw_version`: Hardware revision of the on-chip PON IP module
- `sw_version`: Software revision of the PON library. Here: 72193_D = 0x011A01 = Rev. 1.26.1
- `fw_version_major, fw_version_minor, fw_version_patch`: Firmware revision. Here: Rev. 3.25.0.3.0
- `fw_version_standard`: Identifies the supported PON standard(s):
  - 0: XG-PON (ITU-T G.987), XGS-PON (ITU-T G.9807), NG-PON2 (ITU-T G.989)
  - 1: G-PON (ITU-T G.984)
- `fw_version_is_test`: This value indicates a special debug version and must always be 0 for productive use.

I²C Slave Firmware Version

Note: This is available on PRX126-based systems only.

The binary file is located at `/lib/firmware/prx_i2c_aca_fw.bin`.

The version identifier is located in the first word of the binary and coded as:

- Bit(31:28): Firmware type, must be 1 (I²C slave).
- Bit(27:24): Build ID
- Bit(23:20): Chip type, must be 2.
- Bit(19:16): Chip version, must be 1 or greater
- Bit(15:0): Firmware revision (ascending number)

The binary header is checked by using the `hexdump` tool:

```
# head -c8 /lib/firmware/prx_i2c_aca_fw.bin | hexdump -Cv
00000000  41 43 41 32 10 21 00 18
                      || ||  |   |
                      || ||  +---+--> firmware revision
                      || |+--------> chip version (SoC type)
                      || +---------> chip type
                      |+----------> build ID
                      +-----------> firmware type (I2C slave)
```

The firmware file name in the 10G PON Chipset System Package is constructed from the version identifier, for example:

```
prx_i2c_aca_fw-v1.1.2.0.4.tar.gz
                | | | | |
                | | | | +----------> firmware revision
                | | | +------------> build ID
                | | +--------------> chip type
                | +----------------> chip version (SoC type)
                +------------------> firmware type (I2C slave)
```

PPv4 QoS Firmware Version

The binary file is located at `/lib/firmware/ppv4-qos-fw.elf`.

The version is listed during the boot process, for example:

```
/# logread | grep PP_QOS_FW
Thu Aug 10 14:04:31 2023 kern.info kernel: [   12.903344]
[PP_QOS_FW]:do_load_firmware:316: FW [v 1.16.0] is running soc rev 1)
```

This command also dumps it:

```
# cat /sys/kernel/debug/pp/qos0/geninfo
Driver version: 1.1.0-lgm
FW version:     major(1) minor(16) build(0)
[...]
```

GPHY Firmware Version

The GPHY firmware is located at `/lib/firmware/ltq_fw_PHY31G_IP_prx3xx_Bxx.bin`.

The GPHY firmware version is read through MDIO access:

```
# switch_cli xgmac 3 mdio_rd 0 31 30
xgmac
switch_cli xgmac 3 mdio_rd
OP      CL      DEVADR  PHYID   PHYREG  DATA
=============================================
RD      CL45         0     1F       1E  879D
```


Optical Interface Properties

The optical interface (PMD) parameters are read by generic I²C bus access from the command line, for example using the `i2cdump` command.

```
# i2cdump -f -y 0 0x50
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 03 04 01 00 00 00 00 00 00 00 00 03 63 00 14 c8    ???.........?c.??
10: 00 00 00 00 53 55 50 45 52 58 4f 4e 20 4c 54 44    ....SUPERXON LTD
20: 2e 20 20 20 00 00 00 00 53 4f 47 58 32 36 39 39    .   ....SOGX2699
30: 2d 50 53 47 41 20 20 20 31 30 20 20 04 f6 00 ce    -PSGA   10  ??.?
40: 00 1c 00 00 39 37 39 36 31 31 38 33 32 30 30 32    .?..979611832002
50: 30 35 20 20 31 38 30 38 31 37 20 20 68 f0 02 04    05  180817  h???
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
```

```
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
f0: 34 41 34 31 37 34 36 30 30 30 39 30 30 20 20 20     4A41746000900
```

14.3.2 U-Boot Command Line Functions

Press a key within the configured `bootdelay` time to interrupt the boot process and access the U-Boot command line.

Use `printenv` to list all the U-Boot environment variables or `printenv <name>` to inspect a single one.

Use the `set` command to modify the environment variables. These changes are lost upon a system power down. Call `save` to store such changes permanently.

From the Linux command line, use these commands to access the U-Boot environment variables:

```
# fw_printenv
# fw_setenv
```

14.3.3 PON Command Line Functions

In addition to the standard Linux functions, it is possible to use the CLI to access all PON library functions. For a detailed functional description, refer to the source code of `/pon_lib/include/fapi_pon_<…>.h` or the related documentation derived by Doxygen.

The short forms are available for most of the commands to make manual use easier. However, to avoid ambiguity, it is mandatory to use the full name for some commands. The use of the full names is also recommended in test scripts to improve the readability.

Use these command for a quick help from the CLI to list all functions:

```
# pon
== Group: <CLI commands (misc)> ===
   n/a,           gem_portid_get
   n/a,           gtc_counters_get
   n/a,           ploamd_cfg_set
   n/a,           ploamd_cfg_get
   n/a,           gtc_cfg_set
   n/a,           gtc_cfg_get
   n/a,           psm_cfg_set
   n/a,           psm_cfg_get
   n/a,           psm_counters_get
   n/a,           debug_rogue_stop
   n/a,           debug_rogue_internal_stop
   n/a,           alloc_index_get
   n/a,           debug_trace_stop
   n/a,           aon_cfg_get
   n/a,           alarm_cfg_get
   n/a,           serdes_cfg_set
   n/a,           serdes_cfg_get
   n/a,           gpio_cfg_get
   n/a,           pin_config_set
   n/a,           omci_cfg_get
   n/a,           omci_cfg_set
   1ed,           1pps_event_disable
```

```
1ee,            1pps_event_enable
acg,            alloc_counters_get
acs,            alarm_cfg_set
adcg,           alloc_discard_counters_get
aecg,           auth_enc_cfg_get
aecs,           auth_enc_cfg_set
agpg,           alloc_gem_port_get
aig,            alloc_id_get
alcg,           alarm_limit_cfg_get
alcs,           alarm_limit_cfg_set
asg,            alarm_status_get
ass,            alarm_status_set
atd,            aon_tx_disable
ate,            aon_tx_enable
bg,             bit_get
bs,             bit_set
cg,             cap_get
crg,            cred_get
crs,            cred_set
daa,            debug_alloc_assign
dacg,           debug_alarm_cfg_get
dacs,           debug_alarm_cfg_set
dad,            debug_alloc_deassign
dbes,           debug_bit_error_set
dbpg,           debug_burst_profile_get
dcg,            dp_config_get
dco,            debug_create_omcc
doe,            debug_operational_enter
dpcg,           debug_ploam_cfg_get
dpcs,           debug_ploam_cfg_set
dris,           debug_rogue_internal_start
drng,           debug_random_number_get
drs,            debug_rogue_start
dtcg,           debug_trace_cfg_get
dtcs,           debug_trace_cfg_set
dtpcg,          debug_test_pattern_cfg_get
dtpcs,          debug_test_pattern_cfg_set
dtpd,           debug_test_pattern_disable
dtpe,           debug_test_pattern_enable
dtpsg,          debug_test_pattern_status_get
dtrsg,          debug_trace_run_status_get
dts,            debug_trace_start
dtsg,           debug_trace_status_get
edg,            eeprom_data_get
ercg,           eth_rx_counters_get
etcg,           eth_tx_counters_get
fcg,            fec_counters_get
gacg,           gem_all_counters_get
gcg,            gpon_cfg_get
gpag,           gem_port_alloc_get
gpcg,           gem_port_counters_get
```

```
gpig,           gem_port_index_get
grcs,           gpon_rerange_cfg_set
grsg,           gpon_rerange_status_get
gsg,            gpon_status_get
gtsg,           gpon_tod_sync_get
gtss,           gpon_tod_sync_set
icg,            iop_cfg_get
ics,            iop_cfg_set
lc,             lwi_clear
lcg,            loop_cfg_get
lcs,            loop_cfg_set
ld,             link_disable
le,             link_enable
lg,             limits_get
ls,             lwi_set
ltd,            lwi_test_disable
lte,            lwi_test_enable
ocg,            optic_cfg_get
oig,            omci_ik_get
opg,            optic_properties_get
osg,            optic_status_get
ots,            olt_type_set
pcg,            pqsf_cfg_get
pcs,            pqsf_cfg_set
pd,             psm_disable
pdcg,           ploam_ds_counters_get
pe,             psm_enable
pg,             password_get
pld,            ploam_log_disable
ple,            ploam_log_enable
psg,            ploam_state_get
psmsg,          psm_state_get
ptg,            psm_time_get
pucg,           ploam_us_counters_get
r,              reset
rcg,            req_cfg_get
rcs,            req_cfg_set
rg,             register_get
rig,            registration_id_get
rs,             register_set
sbr,            serdes_biterr_read
sbsta,          serdes_biterr_start
sbsto,          serdes_biterr_stop
scg,            synce_cfg_get
scs,            synce_cfg_set
sd,             synce_disable
se,             synce_enable
shd,            synce_hold_disable
she,            synce_hold_enable
sng,            serial_number_get
ssg,            synce_status_get
```

```
    tcg,            tod_cfg_get
    tcpsg,          twdm_channel_profile_status_get
    tcs,            tod_cfg_set
    tfcg,           twdm_fec_counters_get
    tg,             tod_get
    tocg,           timeout_cfg_get
    tocs,           timeout_cfg_set
    topcg,          twdm_optic_pl_counters_get
    tpdcg,          twdm_ploam_ds_counters_get
    tpucg,          twdm_ploam_us_counters_get
    tsdg,           twdm_sw_delay_get
    tsds,           twdm_sw_delay_set
    tsg,            twdm_status_get
    ttcg,           twdm_tuning_counters_get
    twcg,           twdm_cpi_get
    twcs,           twdm_cpi_set
    twdmcg,         twdm_cfg_get
    txacg,          twdm_xgem_all_counters_get
    txcg,           twdm_xgtc_counters_get
    txpcg,          twdm_xgem_port_counters_get
    ucg,            uart_cfg_get
    ucs,            uart_cfg_set
    um,             user_mngmt
    vg,             version_get
    xcg,            xgtc_counters_get
    xkcs,           xgem_key_cfg_set
== Group: <CLI built-in commands> ===
    help,           Help
    what,           what_string
```

14.3.3.1 PON CLI – Error Codes

When entering an incorrect command, the CLI returns an error. Table 84 lists the possible error codes and their descriptions.

Table 84 PON CLI Error Codes

Error Name	Error Code	Description
PON_STATUS_OK	0	No error
PON_STATUS_OK_NO_RESPONSE	-0101	No ACK or NACK is sent to the firmware by an event, the data handled is only logged.
PON_STATUS_ERR	-1001	A generic or unknown error has occurred.
PON_STATUS_MEM_ERR	-1002	A memory error has occurred.
PON_STATUS_VALUE_RANGE_ERR	-1003	At least one value is out of range.
PON_STATUS_RESOURCE_ERR	-1004	The requested resource is not available.
PON_STATUS_ALLOC_ID_EXISTS	-1005	The allocation ID to be added already exists.
PON_STATUS_ALLOC_ID_MISSING	-1006	The allocation ID to be deleted does not exist.
PON_STATUS_QOS_PORT_INDEX_ERR	-1007	The QoS port index to be assigned is already in use.
PON_STATUS_SUPPORT	-1008	The requested functionality is not supported.
PON_STATUS_FW_NACK	-1009	The firmware message was not acknowledged by the firmware.
PON_STATUS_FW_UNEXPECTED	-1010	The software received an unexpected message from the firmware.
PON_STATUS_MBOX_ERR	-1011	The calling software received an error from the mailbox driver.
PON_STATUS_TIMEOUT	-1012	Did not receive an answer in time.
PON_STATUS_FW_DUP_ERR	-1013	NACK from firmware, because adding of a duplicate entry was rejected.
PON_STATUS_FW_DBG	-1014	NACK from firmware, because PON IP debug mode is disabled.
PON_STATUS_DUPLICATE_GEM_ERR	-1015	Duplicate GEM port entry.
PON_STATUS_FW_CMDERR	-1016	Error in the PON IP message header.
PON_STATUS_NL_ERR	-1017	A netlink operation returned an error.
PON_STATUS_NL_MSG	-1018	An unexpected netlink message was received.
PON_STATUS_INPUT_ERR	-1019	One of the input parameters is wrong.
PON_STATUS_NL_NAME_ERR	-1020	Cannot resolve the pon mailbox netlink name.
PON_STATUS_MBOX_ENOMEM	-1021	Cannot allocate memory.
PON_STATUS_MBOX_EINVAL	-1022	Invalid argument.
PON_STATUS_MBOX_ENODEV	-1023	No such device.
PON_STATUS_DATA_SET_ERR	-1024	Data setting error occurred.
PON_STATUS_GEM_PORT_ID_NOT_EXISTS_ERR	-1025	The GEM port ID does not exist.
PON_STATUS_FW_STATE	-1026	NACK from firmware because the PON IP firmware is in a wrong state.

Table 84 PON CLI Error Codes (cont'd)

Error Name	Error Code	Description
PON_STATUS_TRACE_MODULE_NOT_READY	-1027	The PON IP trace module is not ready.
PON_STATUS_EEPROM_OPEN_ERR	-1028	File open error.
PON_STATUS_EEPROM_SEEK_ERR	-1029	File seek error.
PON_STATUS_EEPROM_READ_ERR	-1030	File read error.
PON_STATUS_EEPROM_WRITE_ERR	-1031	File write error.
PON_STATUS_OPERATION_MODE_ERR	-1032	Operation mode error.
PON_STATUS_LWI_SW_CONTROL_DISABLE	-1033	Local wake-up indication by software control is disabled.
PON_STATUS_BURST_PROFILE_RATE_ERR	-1034	An unexpected burst profile status rate was chosen.
PON_STATUS_MEM_NOT_ENOUGH	-1035	Not enough memory to copy data.
PON_STATUS_ALLOC_GEM_MAP_ERR	-1036	No GEM port mapped to the allocation ID (T-CONT ID) is present.
PON_STATUS_PRBS_DISABLE	-1037	PRBS upstream for optical calibration is disabled.
PON_STATUS_SYNCE_STATE_ERR	-1038	Wrong synchronous Ethernet operation state.
PON_STATUS_MEMCPY_ERR	-1100	Internal error returned by memory copy function.

14.3.3.2 PON CLI – Function Details

The command listing covers all the functions provided by the PON library for all supported PON operating modes. Some functions are only available in a specific PON mode and result in an error response when called in other modes. For example, calling an EPON-only function while the XGS-PON mode is active results in an error.
For information about the command parameters type, use:

```
# pon <command_name>
```

For example:

```
# pon xgtc_counters_get -help <cr>
Long Form: xgtc_counters_get
Short Form: xcg

Output Parameter
- enum fapi_pon_errorcode errorcode
- uint64_t psbd_hec_err_uncorr
- uint64_t psbd_hec_err_corr
- uint64_t fs_hec_err_uncorr
- uint64_t fs_hec_err_corr
- uint64_t lost_words
- uint64_t ploam_mic_err
- uint64_t xgem_hec_err_corr
- uint64_t xgem_hec_err_uncorr
```

The data structures used by the CLI functions reflect the implementation of the PON library and are defined in `pon_lib/include/fapi_pon_<…>.h`.
The functions defined in `fapi_pon_develop.h` are available only when secure debug mode is enabled and return an error when called in normal operation mode.

14.3.3.3 PON CLI – Error Codes

The error codes returned by the CLI functions are listed in `pon_lib/include/fapi_pon_error.h` or the related documentation derived by Doxygen.

14.3.3.4 PON CLI – Optical Interface Status (up to Release 1.16.x)

The `pon optic_status_get` CLI function retrieves data from the PMD device over the I^2C interface. Therefore, the command requires the paths to the I^2C devices to be specified. For example:

```
# pon optic_status_get /sys/bus/i2c/devices/0-0050/eeprom /sys/bus/i2c/devices/0-
0051/eeprom
errorcode=0 temperature=11999 voltage=32999 bias=0 tx_power=-20000 rx_power=-20000
rx_los=0 tx_disable=0 tx_fault=0
```

Note: Both EEPROM file paths must be given in the command.

The required EEPROM names can be retrieved by calling

The temperature is provided as a signed value given in a 32-bit signed 2s complement format, in increments of 1/256 Celsius degrees. The output number range is from -32767 to 32767. The physical temperature range is from -128 °C to +128 °C.

```
# uci show optic.sfp_eeprom
optic.sfp_eeprom=optic
optic.sfp_eeprom.comment='settings for pon v1.17 or later'
optic.sfp_eeprom.dmi='/sys/class/pon_mbox/pon_mbox0/device/eeprom51'
optic.sfp_eeprom.serial_id='/sys/class/pon_mbox/pon_mbox0/device/eeprom50'
```

The transmit power and the receive power are given as signed values in a 2s complement integer format, in dBm (referred to 1 mW). The granularity is 0.002 dB. The valid value number range is from -20000 to 4082. To get dBm, divide this value by 500. This represents physical values from -40 dBm (0.1 uW) to +8.164 dBm (6.5535 mW). A transmit power of 0 mW is reported as -32767.

The voltage is given as an unsigned 32-bit integer value with an LSB representing a step of 100 uV. The valid number range is from 0 to 65535, representing physical values from 0 V to 6.5535 V.

14.3.3.5 PON CLI – Optical Interface Status (Release 1.17.0 or Later)

The `pon optic_status_get` CLI function retrieves data from the PMD device over the I^2C interface. Therefore, the command requires the paths to the I^2C devices to be specified. For example:

```
# pon optic_status_get /sys/class/pon_mbox/pon_mbox0/device/eeprom50
/sys/class/pon_mbox/pon_mbox0/device/eeprom51
errorcode=0 temperature=11999 voltage=32999 bias=0 tx_power=-20000 rx_power=-20000
rx_los=0 tx_disable=0 tx_fault=0
```

Note: Both EEPROM file paths must be given in the command.

The required EEPROM names can be retrieved by calling

```
# uci show optic.sfp_eeprom
optic.sfp_eeprom=optic
optic.sfp_eeprom.dmi='/sys/class/pon_mbox/pon_mbox0/device/eeprom51'
optic.sfp_eeprom.serial_id='/sys/class/pon_mbox/pon_mbox0/device/eeprom50'
```

The temperature is provided as a signed value given in a 32-bit signed 2s complement format, in increments of 1/256 Celsius degrees. The output number range is from -32767 to 32767. The physical temperature range is from -128 °C to +128 °C.

The transmit power and the receive power are given as signed values in a 2s complement integer format, in dBm (referred to 1 mW). The granularity is 0.002 dB. The valid value number range is from -20000 to 4082. To get dBm,

divide this value by 500. This represents physical values from -40 dBm (0.1 uW) to +8.164 dBm (6.5535 mW). A transmit power of 0 mW is reported as -32767.

The voltage is given as an unsigned 32-bit integer value with an LSB representing a step of 100 µV. The valid number range is from 0 to 65535, representing physical values from 0 V to 6.5535 V.

Another option to retrieve status information from the optical interface by using the `ethtool` command is described in **Chapter 14.4.7** (**Ethernet Interface Status on ANI side – ethtool**).

14.3.3.6 PON CLI – PON Optical Transmitter and Receiver Testing

The PON-side SerDes provides test functions to send and receive test signals. It provides a 64-bit cumulative bit error counter and the measurement time information in ms from the measurement start.
These functions are provided:

- `debug_test_pattern_cfg_set`
  Configures the test pattern to be transmitted (`tx_type`) and expected to be received (`rx_type`).
  The same pattern is used for transmission and reception.
  – 1: PRBS31_28 - PRBS pattern using the polynomial X^31 + X^28 + 1.
  – 2: PRBS23_18 - PRBS pattern using the polynomial X^23 + X^18 + 1.
  – 3: PRBS23_21 - PRBS pattern using the polynomial X^21 + X^16 + X^8 + X^5 + X^2 + 1
  – 4: PRBS16 - PRBS pattern using the polynomial X^16 + X^5 + X^4 + X^3 + 1.
  – 5: PRBS15 - PRBS pattern using the polynomial X^15 + X^14 + 1.
  – 6: PRBS11 - PRBS pattern using the polynomial X^11 + X^9 + 1.
  – 7: PRBS9 - PRBS pattern using the polynomial X^9 + 1.
  – 8: PRBS7 - PRBS pattern using the polynomial X^7 + X^6 + 1.
  – 9: Repetitive fixed 8-bit test pattern. The pattern is configurable.
  – 10: Repetitive fixed 8-bit test pattern, every second pattern is inverted to make the signal DC-balanced.
  – 11: Mixed pattern mode.This sequence is repeated:0x00 - <8-bit pattern> - 0x3FF - <inverted 8-bit pattern> - 0x00 - …
  Note: Depending on the external test pattern generator in use, the PRBS test patterns may require to be inverted at the test pattern generator output.
  The command allows to invert the transmitted and received pattern independently (`rx_inv`, `tx_inv`)
- `debug_test_pattern_cfg_get`
  Reads the test pattern configuration back.
- `debug_test_pattern_enable`
  Starts sending the configured test pattern on a selected upstream wavelength channel. Before sending this command, the test pattern must be configured by using `debug_test_pattern_cfg_set`. These operation modes are available:
  – 1: RX, switches the test pattern checker on.
  – 2: TX_LOOP_TIMED, switches the test pattern transmission with loop timing. A suitable receive signal for the CDR is required.
  Note: The presence of a downstream optical signal is checked by reading the optical interface rx_los status from the PMD device by calling:
  ```
  # pon optic_status_get /sys/bus/i2c/devices/0-0050/eeprom
  /sys/bus/i2c/devices/0-0051/eeprom
  ```
  – 3: TX_LOCAL_TIMED, switches the test pattern transmission with the local clock; a valid receive signal is not required.
  – 4: TX_RX: Test pattern reception and transmission is switched on with loop timing; the transmit pattern is clocked with the recovered clock from the received signal.
  – 5: TX_TO_RX_LOOP, the test pattern transmitter is looped back to the test pattern checker.
  – 6: RX_TO_TX_LOOP_PMD, the received data is looped back to the remote end.
  – 7: RX_TO_TX_LOOP_PON

- `debug_test_pattern_disable`
  Stops sending the test pattern.
- `serdes_biterr_start`
  – Starts a bit error measurement on the received test pattern.
  – The counter is automatically set to 0 and the start time is internally stored.
  – When a measurement is already running, it is automatically stopped and restarted.
  – When the counter is running, it is restarted which resets the counter.
- `serdes_biterr_stop`
  – Stops the bit error measurement.
  – The counter result is returned together with the time which has expired since the measurement was started.
  – When the counter has already been stopped (manually or automatically due to a too high bit error rate), an error is returned (`errorcode=-1001 counter=0 period_ms=0`).
- `serdes_biterr_read`
  – Reads the current counter value and measurement time.
  – The counter continues counting and the time continues to increment with every read interval.
  Note: The SerDes counter polling interval is about 400 ms. The interval is defined in drv_pon_mbox.c as #define SERDES_BIT_ERROR_COUNTER_POLLING_INTERVAL 400.
  – To see whether the counter is running, read it and check whether the time increases.
- `debug_bit_error_set`
  Inserts a single bit error into the transmitted test pattern.

Attention: Because sending of a test pattern breaks a live PON system, the usage of the test pattern transmission is possible only when debug mode is enabled. It must only be used in a setup where the ONU under test does not disturb the traffic of other ONUs connected to the same OLT.

These commands must be sent in the correct sequence for a receive bit error rate measurement:

- `pon debug_test_pattern_cfg_set <pattern type> <fixed pattern>`
  The value for the fixed pattern must always be provided. Use 0 when a PRBS pattern is selected by the first variable.
- `pon debug_test_pattern_enable <test_mode> <wl_ch_id>`
  Enables the selected test mode with the previously configured test pattern on a selected upstream wavelength channel. On single-wavelength systems use the value 0.
- `pon serdes_biterr_start`
  Starts the receive pattern synchronization and resets the bit error counter.
- `pon serdes_biterr_read`
  Intermediate bit error readouts is done. Each readout provides the number of counted bit errors and the time since the start of the measurement (in ms). This allows to calculate the bit error rate, based on the known receive data rate.
- …
- `pon serdes_biterr_read`
- `pon serdes_biterr_stop`
  Stops the measurement, returns the last counter value and stops the counting.
  Do not send the stop command when the counting was already stopped automatically due to a too high bit error rate. Otherwise, an error is reported (`errorcode=-1001`).
- `pon debug_test_pattern_disable`
  Stops sending and receiving test patterns.
- `pon serdes_biterr_read`
  The last counter value is read at any time after the counting was stopped.

Notes

1. *When the hardware counter overruns due to a too high bit error rate (> 1e-3 at 10 Gbit/s), the counting is stopped and a counter value of 2^64-1 is returned (counter=18446744073709551615). This indicates that the counter value is invalid.*
2. *To change the test pattern, the test must be stopped before (pon debug_test_pattern_disable). Otherwise, an error is reported.*
3. *To start a test, the test pattern must be defined first (pon debug_test_pattern_cfg_set <…> <…>). Otherwise, an error is reported.*
4. *When the counting is running, call the start command to restart a bit error measurement. The counter is reset and the new start time is used to calculate the measurement duration.*

14.3.3.7 PON CLI – PLOAM Activation State Machine Testing

There are two functions to perform tests on the PLOAM activation state machine by forcing a re-ranging through the OLT:

- `gpon_rerange_cfg_set`
- `gpon_rerange_status_get`

These tests are available:

- Simulate the loss of the downstream synchronization (LODS) for a configurable time (from 375 µs to1 s).
- Force the PLOAM activation state machine into state O1.1.
- Disable the laser transmitter to simulate the loss of the transmit signal.

Example: Simulate loss of downstream synchronization for a short time.

Simulate LODS for 1 ms.

```
# pon gpon_rerange_cfg_set 8 0 0 0
errorcode=0
# pon gpon_rerange_status_get
errorcode=0 lods_time=0 active=0 psmdis=0 txdis=0
```

Expect to move to PLOAM state O6 and then back to O5.x.

Example: Simulate loss of downstream synchronization for a long time.

Simulate LODS for 1 s.

```
# pon gpon_rerange_cfg_set 8000 0 0 0
errorcode=0
# pon gpon_rerange_status_get
errorcode=0 lods_time=0 active=0 psmdis=0 txdis=0
```

Expect to move to PLOAM state O1.1 and then back to O5.x.

Example: Simulate loss of the operational PLOAM state.

```
# pon gpon_rerange_cfg_set 3 0 1 0
errorcode=0
# pon gpon_rerange_status_get
errorcode=0 lods_time=0 active=0 psmdis=1 txdis=0
```

Expect to lose O5.x state into O1.1

```
# pon gpon_rerange_cfg_set 3 0 0 0
errorcode=0
# pon gpon_rerange_status_get
```

```
errorcode=0 lods_time=0 active=0 psmdis=0 txdis=0
```

Expect to regain O5.x state after some time (the timing depends on OLT).

Example: Simulate loss of the transmit signal.

```
# pon gpon_rerange_cfg_set 3 0 0 1
errorcode=0
# pon gpon_rerange_status_get
errorcode=0 lods_time=0 active=0 psmdis=0 txdis=1
```

Expect to lose O5.x state.

```
# pon gpon_rerange_cfg_set 3 0 0 0
errorcode=0
# pon gpon_rerange_status_get
errorcode=0 lods_time=0 active=0 psmdis=0 txdis=0
```

Expect to regain O5.x state after some time (the timing depends on OLT).

14.3.3.8 PON CLI – GEM Port Counters

To inspect the traffic on the PON, the GEM (XGEM) counters are read per individual GEM/XGEM port ID.

```
root@prx321-sfu-qspi-pon:# pon gem_port_counters_get 1
errorcode=0 gem_port_id=1 tx_frames=1046 tx_fragments=533 tx_bytes=24624
rx_frames=513 rx_fragments=0 rx_bytes=24624
key_errors=0
```

GEM/XGEM frames and fragment are counted separately. In the previous example, on GEM port ID 1, there are:

- 24624 bytes in 1046 upstream GEM/XGEM frames, this number includes 533 GEM/XGEM frames which are only fragments (not the last GEM/XGEM frame of a set of fragments). So in total we have transmitted 1046 - 533 = 513 Ethernet frames.
- 24624 bytes in 513 downstream GEM/XGEM frames, from which none has been fragmented. So we have received also 513 Ethernet frames.

14.3.3.9 PON CLI – Allocation Counters

These two counters allow inspection of the upstream data allocations:

- `pon alloc_counters_get`
  - This counter reports the number of allocations, the number of GEM Idle frames, and the average bandwidth per allocation index. This is useful to check the number and size of assigned upstream time slots.
- `pon alloc_discard_counters_get`
  - This counter reports the number of allocations that could not be served due to different reasons. The counter details are shown in **Table 85** and **Table 86**.

Table 85 Allocation Discard Counters

Discard Counter Number	Discard Reason
0	Number of skipped allocations due to bandwidth map burst errors.
1	Number of foreign allocations in the bandwidth map.
2	Number of own allocations in foreign bursts.
3	Number of foreign allocations in no bursts.
4	Number of foreign allocations as start of bursts.

Table 85 Allocation Discard Counters (cont'd)

Discard Counter Number	Discard Reason
5	Number of bandwidth map HEC errors.
6	Number of times that more than 64 allocations have been received.
7	Number of HLend HEC errors.
Others	Reserved

Table 86 Allocation Rule Counters

Rule Counter Number	Discard Rule
0	Reserved
1	Reserved
2	Number of times that the spacing of adjacent bursts has been violated.
3	Reserved
4	Number of times that the start time has exceeded a value of 9719 maximum start time.
5	Reserved
6	Number of times that the number of allocation structures has exceeded a value of 16 per single allocation.
7	Number of times that the number of allocation structures has exceeded a value that can be served by the ONU in a single upstream burst.
8	Number of times that the number of burst allocation series has exceeded a value of 16 for this ONU.
9	Number of times that the maximum grant size has been exceeded.
10	Number of times that the maximum FS burst size has been exceeded.
11	Number of times that this rule has not been met: BurstStartTime + (sum of GrantSize) <= 14580
12	Number of times that no allocations were found in the current frame after downstream time 13.168 us has passed and the downstream FIFO is still empty.
13	Number of times that the firmware unexpectedly missed to read all downstream allocations in time.
14	Number of times that the upstream allocation FIFO is not ready to receive the next burst.
15	Number of unknown profiles received.
16	Number of allocations successfully written into the upstream allocation FIFO.
Others	Reserved

14.3.3.10 PON CLI – Hexadecimal Output

The numbers returned by the PON CLI functions are represented as decimal values. When the result must be returned as hexadecimal values, an automatic conversion is done by adding `| tohex` to the command.

For example:

```
# pon version_get
```

```
errorcode=0 hw_version=7 sw_version=71176 fw_version_major=3 fw_version_minor=21
fw_version_standard=0 fw_version_platform=3 fw_version_patch=14
fw_version_is_test=0 fw_timestamp=1669214605

# pon version_get | tohex
errorcode=0x0 hw_version=0x7 sw_version=0x11608 fw_version_major=0x3
fw_version_minor=0x15 fw_version_standard=0x0 fw_version_platform=0x3
fw_version_patch=0xE fw_version_is_test=0x0 fw_timestamp=0x637E318D
```

14.3.3.11 PON CLI – Dying Gasp Check

To check the external hardware that controls the Dying Gasp input voltage, the internal Dying Gasp status can be read from the PON hardware module.

Normal operation:

If bit 15 is set, the supply voltage is too low and the Dying Gasp detection is active.

Examples:

```
# ponmbox -c 0xe1 0x00001 0x00003fed
errorcode=0 recv_msg_length=12 message=00000001 00003fed 00004**4**00a
```

Low supply voltage:

```
# ponmbox -c 0xe1 0x00001 0x00003fed
errorcode=0 recv_msg_length=12 message=00000001 00003fed 0000**c**00a
```

14.3.4 OMCI Command Line Functions

To restart the OMCI daemon manually, use:

```
/etc/init.d/omcid.sh restart
```

To stop the OMCI daemon, use:

```
/etc/init.d/omcid.sh stop
```

To start the OMCI daemon again, use:

```
/etc/init.d/omcid.sh start
```

To display all OMCI managed entities created in the active MIB, use:

```
# omci md > omci_md.txt
```

Use this command for a more detailed analysis. It prints a list of all managed entities together with all attribute values.

```
# omci mda > omci_mda.txt
```

Note: Only the managed entities and attributes which are active according to the selected OMCC version are listed.

Alternatively, the list is dumped in XML format to be read by a spreadsheet program using:

```
# omci mdx > omci_mdx.xml
```

To list the single managed entities, use:

```
# omci meg <class> <instance>
```

To print a complete list of debug commands, use:

```
# omci help
== Group: <CLI commands (misc)> ===
   aas,            attr_avc_send
   ac,             attr_change
   asnr,           alarm_seq_num_reset
   atg,            action_timeout_get
   ats,            action_timeout_set
   cda,            class_dump_all
   cdx,            class_dump_xml
   cg,             class_get
   cpg,            class_prop_get
   dlg,            dbg_level_get
   dls,            dbg_level_set
   dmlg,           dbg_module_level_get
   dmls,           dbg_module_level_set
   fe,             failsafe_enable
   ie,             interval_end
   img,            iop_mask_get
   ims,            iop_mask_set
   mcg,            msg_counters_get
   md,             mib_dump
   mda,            mib_dump_all
```

```
    mdce,           msg_dump_console_enable
    mdd,            msg_dump_disable
    mdfe,           msg_dump_file_enable
    mdx,            mib_dump_xml
    meadg,          managed_entity_attr_data_get
    meads,          managed_entity_attr_data_set
    meag,           managed_entity_alarm_get
    meaog,          managed_entity_attr_offset_get
    meapg,          managed_entity_attr_prop_get
    meas,           managed_entity_alarm_set
    measg,          managed_entity_attr_size_get
    meatg,          managed_entity_attr_type_get
    meavg,          managed_entity_attr_version_get
    mec,            managed_entity_create
    mecg,           managed_entity_count_get
    med,            managed_entity_delete
    meg,            managed_entity_get
    meicg,          managed_entity_inst_count_get
    meis,           managed_entity_is_supported
    mng,            msg_num_get
    mps,            msg_pool_size
    mr,             mib_reset
    ms,             mib_store
    pd,             processing_disable
    pdmlg,          pa_dbg_module_level_get
    pdmls,          pa_dbg_module_level_set
    pe,             processing_enable
    rmr,            raw_message_recv
    rms,            raw_message_send
    s,              status
    swiu,           sw_image_upgrade
    tcici,          tci_check_ignore
    vig,            version_information_get
== Group: <CLI built-in commands> ===
    help,           Help
    quit,           Quit
```


For more details, see **Section 4.2**, **OMCI Daemon**.

14.3.4.1 OMCI Debug Level Settings

The debug level for console output is set by:

```
# omci dmls <module> <level>
```

- Module selection
  - OMCI_DBG_MODULE_NO = 0
  - OMCI_DBG_MODULE_CORE = 1
  - OMCI_DBG_MODULE_API = 2
  - OMCI_DBG_MODULE_MIB = 3
  - OMCI_DBG_MODULE_ME = 4
  - OMCI_DBG_MODULE_MSG_DUMP = 5
  - OMCI_DBG_MODULE_PM = 6

  - OMCI_DBG_MODULE_MIB_RESET = 7
  - OMCI_DBG_MODULE_IFXOS = 8
  - OMCI_DBG_MODULE_VOIP = 9
  - OMCI_DBG_MODULE_MCC = 10
  - OMCI_DBG_MODULE_UBUS = 11
- Level selection
  - OMCI_DBG_MSG = 0
  - OMCI_DBG_PRN = 1
  - OMCI_DBG_WRN = 2
  - OMCI_DBG_ERR = 3
  - OMCI_DBG_OFF = 4

The debug levels on OMCI level are defined as:

- 0: MSG, message level
  - Prints out any debug message provided by the code.
    This leads to a large CPU load and disturbs at least parts of the system behavior.
    Note: Usage of this level is not recommended. This most probably leads to a system misbehavior (up to a system hangup), due to the tight memory constraints.
- 1: PRN, print level
  - Prints additional information, but not a complete dump.
- 2: WARN, warning level
  - Prints out warnings and errors.
- 3: ERR, error level
  - Only prints out error messages.
    This is the recommended default setting.
- 4: OFF, no printout.

14.3.4.2 OMCI Message Dump

Use this command to log all OMCI messages exchanged between OLT and ONU into a file under Linux:

```
tcpdump -i gem-omci -w /tmp/omci.pcap
```

The resulting file is copied to a PC and viewed with an analysis tool, such as Wireshark.

To log the OMCI activities on the command line, use

```
omci message_dump_console_enable
```

On RDK-B systems this log is stored in the journald file system. Retrieve the log by entering this command:

```
journalctl -u omcid --no-pager
```

14.3.4.3 PON Adapter Debug Level Settings

To set the debug output level for the PON adapter, use:

```
# omci pa_dbg_module_level_set
Long Form: pa_dbg_module_level_set
Short Form: pdmls

Input Parameter
- char pa_ll_module[20] - module index or name
- uint8_t dbg_level
   0: messages
   1: prints
   2: warnings
   3: errors
```

```
   4: off

Output Parameter
- enum omci_error errorcode

Loaded modules:
0. pon_adapter
1. libpon
2. libponnet
3. libponimg
```

The explicit module name must be given in the command, as listed by the help output under “Loaded modules”.

The current debug level can be checked by calling

```
# omci pa_dbg_module_level_get pon_adapter
errorcode=0 dbg_level=3
# omci pa_dbg_module_level_get libpon
errorcode=0 dbg_level=2
# omci pa_dbg_module_level_get libponnet
errorcode=0 dbg_level=3
# omci pa_dbg_module_level_get libponimg
errorcode=0 dbg_level=3
```

For example:

```
# # omci pa_dbg_module_level_set libponnet 3
errorcode=0
# omci pa_dbg_module_level_get libponnet
errorcode=0 dbg_level=3
```


14.3.4.4 OMCC Version

The OMCC version is checked by reading the OMCC version attribute of the ONU2-G managed entity:

```
# omci meadg 257 0 2
errorcode=0 attr_data=a3
```

14.3.4.5 Manual Managed Entity Creation

It is possible to create managed entities for testing purposes. This is also done while connected to an OLT, as long as the new one is not supposed to be handled by the OLT. For example to create another T-CONT, use:

```
# omci managed_entity_create 262 0x8010 0xFFFF 1 1
```

The syntax is the same as the MIB initialization files. When a required single argument is composed of multiple values, use a slightly different input format. For example, create the Ethernet counters as in:

```
# omci managed_entity_create 334 1 0
"\xff\xff\x00\x0b\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
```

14.3.4.6 OMCI Counter Read-out

The OMCI counters provide two sets of values, the current counter status and the 15-minute history. Both are read on the command line. For example:

```
omci managed_entity_get 334 1 0
```

reads the current counter status while

```
omci managed_entity_get 334 1 0 1
```

reads the historical values. Note the appended `1` in the command.

14.3.4.7 OMCI Extended VLAN Rule Decoding

A Python script is available at to decode the hexadecimal view of an Extended VLAN rule that is generated by the OMCI debug command into a readable JSON format. This script is located at

`/pon_net_lib/scripts/extvlan.py`

For example:

```
$ ./extvlan.py --omci-hex "0xF8 0x00 0x00 0x00 0x80 0x00 0x00 0x01 0x40 0x0F 0x00
0x00 0x00 0x03 0x0F 0xAE"
{
    "FilterOuterPriority": 15,
    "FilterOuterVID": 4096,
    "FilterOuterTPID": 0,
    "FilterInnerPriority": 8,
    "FilterInnerVID": 0,
    "FilterInnerTPID": 0,
    "FilterEtherType": 1,
    "TreatTagsToRemove": 1,
    "TreatOuterPriority": 15,
    "TreatOuterVID": 0,
    "TreatOuterTPID": 0,
    "TreatInnerPriority": 3,
    "TreatInnerVID": 501,
    "TreatInnerTPID": 6
}
```

It also allows the creation of the hexadecimal notation from a command line input, such as:

```
$ ./extvlan.py --meid 257 --fop 15 --fov 4096 --fot 0 --fip 15 --fiv 4096 --fit 0 -
-fe 0 --ttr 0 --top 15 --tov 0 --tot 0 --tip 3 --tiv 333 --tit 6
omci meads 171 257 6 0xF8 0x00 0x00 0x00 0xF8 0x00 0x00 0x00 0x00 0x0F 0x00 0x00 0x00
0x03 0x0A 0x6E
```

This can be used to create Extended VLAN rules manually for system debugging.

14.3.4.8 OMCI Error Codes

Table 87 lists the OMCI CLI functions return error codes.

Table 87 OMCI Error Codes

OMCI Error Code	Error Cause
1	Resource temporarily unavailable
0	Success, no error
-1	Unspecified error
-2	Not found
-3	Not available
-4	No memory
-5	Not supported
-6	No data
-7	Configuration mismatch
-8	Resource already exists
-9	Resource is not available

Table 87 OMCI Error Codes (cont'd)

OMCI Error Code	Error Cause
-10	Resource not found
-11	Match not found
-12	Invalid value
-13	Driver error
-14	OMCI message FIFO is full
-15	Managed entity ID is invalid
-16	Managed entity ID not found
-17	Managed entity already exists
-18	OMCI action error
-19	Managed entity is not supported
-20	Attribute is invalid
-21	Managed entity action is invalid
-22	Locking error
-23	Invalid pointer
-24	Out-of-bounds error
-25	Interface not found
-26	Memory access error
-27	Invalid TCI in OMCI message

14.3.5 Debug Level Definition

The debug levels on driver level are defined as:

- 0: MSG, message level
  - Prints out any debug message provided by the code.
    This leads to a large CPU load and disturbs at least parts of the system behavior.
    Note: Usage of this level is not recommended.
- 1: WARN, warning level
  - Prints out warnings and errors.
- 2: ERR, error level
  - Only prints out error messages. This is the recommended default setting.
- 3: OFF, no printout.

14.3.6 PON Debugging with pond

The PON debug daemon `pond` provides access to basic configurations and allows PLOAM message and event status logging. Check the built-in help for details.

```
# pond -h
pond: PON daemon
--firmware    -f  <value>
--reset       -r
--help        -h
--mac_sa      -s  <value>
--mac_da      -d  <value>
--serial      -n  <value>
--regid       -i  <value>
--identifier  -o  <value>
--tod only    -t
--verbose     -v
--mode        -m  <value>
```

Example to log the PLOAM messages and state transitions:

```
pond -v &
```


Notes

1. *The command "pond --reset" is no longer supported. Use "pon reset 0" instead*
2. *The `pond` functionality is only available when the secure debug mode is enabled for the ONU.*

14.3.7 PON Status Debugging with pontop

The `pontop` function provides a quick system overview of the PON interface.

Using the `pontop` command starts the application and provides a quick system overview of the PON interface. These command line options are used with the command:

- `pontop -h | -help | --help | help | h`
  Provides an overview of the command line options.
- `pontop <command>`
  Any command listed by the help function is used as command line option.
  A single page is displayed and the program returns to the command line.
- `pontop`
  When given without a command line option, the `ptop` status page appears. From here any of the single-/dual-character commands (as shown by the help page) are entered to display one of the other information screens.

The `pontop` help screen appears on the next page. Some of the options are not available for all PON types and are only displayed and accessible when meaningful.

Display of pontop

```
Help

Up, Ctrl-y      Scroll up                    Pg up, Ctrl-u   Scroll page up
Down, Ctrl-e    Scroll down                  Pg down, Ctrl-d Scroll page down
Home            Jump to first line           End             Jump to last line
/               Define filter                Enter           Drop group key
```

```
Ctrl-w          Write selected (current page) counters to file
                                                /tmp/<Date>_<Time>_<Group>.txt
Ctrl-a          Dump all pages to file /tmp/<Date>_<Time>.txt

Ctrl-x, Ctrl-c  Exit program
```

?	Help	s	Status
c	Capability and Configuration	l	LAN Interface Status & Counters
w	Active alarms	g-s	GEM/XGEM Port Status
g-c	GEM/XGEM Port Counters	g-d	GEM/XGEM port DS Counters
g-u	GEM/XGEM port US Counters	e-d	GEM/XGEM port Eth DS Cnts
e-u	GEM/XGEM port Eth US Cnts	f	FEC Status & Counters
t	GTC/XGTC Status & Counters	p-s	Power Save Status
p-c	PSM Configuration	a-c	Allocation Counters
p-d	PLOAM Downstream Counters	p-u	PLOAM Upstream Counters
o-s	Optical Interface Status	o-i	Optical Interface Info
d-b	Debug Burst Profile	C-o	CQM ofsc
C-q	CQM Queue Map	d-p	Datapath Ports
d-q	Datapath QOS	P-b	PPv4 Buffer MGR HW Stats
P-p	PPv4 QoS Queue PPS	P-s	PPv4 QoS Queues Stats
P-t	PPv4 QoS Tree	P-q	PPv4 QoS QStats

14.3.7.1 pontop CLI Functions

Instead of displaying the life update of status and counters the `pontop` function is used in scripts to collect status and counter data snapshots. These options are provided:

```
# pontop -h
pontop V-1Usage: pontop [options]

Options:
        -b, --batch       Start in `Batch mode`
        -d, --delay <ms>  Counters update delay
        -g, --group <grp> Show specified counters upon startup
                          Possible group values:
                             ? - Help
                             s - Status
                             c - Capability and Configuration
                             l - LAN Interface Status & Counters
                             w - Active alarms
                             GEM/XGEM Port Status
                             GEM/XGEM Port Counters
                             GEM/XGEM port DS Counters
                             GEM/XGEM port US Counters
                             GEM/XGEM port Eth DS Cnts
                             GEM/XGEM port Eth US Cnts
                             f - FEC Status & Counters
                             t - GTC/XGTC Status & Counters
                             Power Save Status
                             PSM Configuration
                             Allocation Counters
                             PLOAM Downstream Counters
                             PLOAM Upstream Counters
                             Optical Interface Status
                             Optical Interface Info
                             Debug Burst Profile
                             CQM ofsc
                             Datapath Ports
                             Datapath Lookup
                             Datapath QOS
                             PPv4 Buffer MGR HW Stats
                             PPv4 QoS Queue PPS
                             PPv4 QoS Queues Stats
                             PPv4 QoS Tree
                             PPv4 QoS QStats

        -h, --help        Print help (this message)
        -v, --version     Print version information
```


Store to File

The command `pontop -b` stores the current status of all sub-pages into the file `/tmp/pontop.txt`.

```
# pontop -b
Saved dump to /tmp/pontop.txt
```

Show a Single Sub-page

To show a selected sub-page when the `-g` option is used. For example, to show the status sub-page or capabilities:

```
# pontop -b -g s
Page: Status
OPTION                                          VALUE
PON IP HW version                              : 7
PON IP FW version                              : 3.21.0.3.14-1669214605
PON IP SW version                              : 1.22.8
PON IP pontop version                          : 1.7.2

SW uptime                                      : t.b.d

PON type active                                : t.b.d

PON capability                                 : 000731f9
   GEM Ports                                   : 256

PON PLOAM Status                               : O1.1, Initial off-sync state

FEC upstream                                   : OFF
FEC downstream                                 : OFF

ONU Authentication Status                      : 0

OC Structure contents
---------------------
PON ID                                         : 0x00000000
TOL                                            : -30.0 dBm
RE                                             : No reach extender present
ODN Class                                      : N1
DS FEC                                         : Downstream FEC disabled
P                                              : TC mode is G.987.3
Link Type                                      : Unspecified
Administrative Label                           : 0x00000000
DWLCH ID                                       : 0x0
R                                              : value not available
C                                              : value not available

# pontop -b -g c
Page: Capability and Configuration
OPTION                                          VALUE
Serial number                                  : <value>
Registration ID                                : <value>

Capability details
------------------
Basic mode(s)                                  : G.987|G.989|G.9807|
OMCI support                                   : OMCI Annex E|F
Power saving mode(s)                           : doze|cyclic sleep|watchful …|
DBA mode(s)                                    : mode0|mode1
```

```
Crypto mode(s)                                    : AES-CMAC-128|

GEM Ports                                         : 256
Allocations                                       : 64
Debug mode                                        : enabled
```

```
# pontop -b -g w
Page: Active alarms
Alarm type       Alarm                      Description
LEVEL            PON_ALARM_STATIC_LOS       Loss of signal
LEVEL            PON_ALARM_STATIC_LODS      Loss of downstream synchronization
LEVEL            PON_ALARM_STATIC_ERR_LOSS Same signal as LOS just level sensitive
```

Note: Some events may cause multiple alarms to be triggered.

14.3.8 Ethernet Switch Command Line Functions

The Ethernet switch hardware (GSWIP, Gigabit Switch IP) provides a set of CLI functions to access the configuration and read counters.

For a quick help from the command line to list all functions type, use:

```
# switch_cli <cr>
```

For information about the command parameters, type:

```
***** Switch CLI Help Menu *****

Select any of the below Switch Feature Operations, to display the related Switch CLI
Commands

0.  Quit
1.  RMON Operation
2.  MAC  Operations
3.  VLAN Operation
4.  Extended VLAN Operations
5.  VLAN Filter Operations
6.  Bridge Port and Bridge Operations
7.  PCE Operations
8.  QOS Operations
9.  8021x Operations
10. Multicast Operations
11. Trunking Operations
12. WOL Operations
13. MonitorPort Operations
14. PMAC Operations
15. PAE Operation
16. MDIO Operation
17. MMD operations
18. MAC Filter Operation
19. Debug Operation
20. Xgmac Operation
21. GSW Sub-system Operation
22. Display All Switch Operation

Enter the option (0 to 22) to continue :
```


A number from 0 to 22 is entered to check for a dedicated subset of functions or `22` to list all functions:

```
1. GSW_8021X_EAPOL_RULE_GET
2. GSW_8021X_EAPOL_RULE_SET
3. GSW_8021X_PORT_CFG_GET
4. GSW_8021X_PORT_CFG_SET
5. GSW_CAP_GET
6. GSW_CFG_GET
7. GSW_CFG_SET
8. GSW_CPU_PORT_CFG_GET
9. GSW_CPU_PORT_CFG_SET
10. GSW_CPU_PORT_EXTEND_CFG_GET
11. GSW_CPU_PORT_EXTEND_CFG_SET
12. GSW_DISABLE
```

13. GSW_ENABLE
14. GSW_HW_INIT
15. GSW_MAC_TABLE_CLEAR
16. GSW_MAC_TABLE_ENTRY_ADD
17. GSW_MAC_TABLE_ENTRY_QUERY
18. GSW_MAC_TABLE_ENTRY_READ
19. GSW_MAC_TABLE_ENTRY_REMOVE
20. GSW_MDIO_CFG_GET
21. GSW_MDIO_CFG_SET
22. GSW_MDIO_DATA_READ
23. GSW_MDIO_DATA_WRITE
24. GSW_MMD_DATA_READ
25. GSW_MMD_DATA_WRITE
26. GSW_MONITOR_PORT_CFG_GET
27. GSW_MONITOR_PORT_CFG_SET
28. GSW_MULTICAST_ROUTER_PORT_ADD
29. GSW_MULTICAST_ROUTER_PORT_READ
30. GSW_MULTICAST_ROUTER_PORT_REMOVE
31. GSW_MULTICAST_SNOOP_CFG_GET
32. GSW_MULTICAST_SNOOP_CFG_SET
33. GSW_MULTICAST_TABLE_ENTRY_ADD
34. GSW_MULTICAST_TABLE_ENTRY_READ
35. GSW_MULTICAST_TABLE_ENTRY_REMOVE
36. GSW_PORT_CFG_GET
37. GSW_PORT_CFG_SET
38. GSW_PORT_LINK_CFG_GET
39. GSW_PORT_LINK_CFG_SET
40. GSW_PORT_PHY_ADDR_GET
41. GSW_PORT_PHY_QUERY
42. GSW_PORT_REDIRECT_GET
43. GSW_PORT_REDIRECT_SET
44. GSW_PORT_RGMII_CLK_CFG_GET
45. GSW_PORT_RGMII_CLK_CFG_SET
46. GSW_QOS_CLASS_DSCP_GET
47. GSW_QOS_CLASS_DSCP_SET
48. GSW_QOS_CLASS_PCP_GET
49. GSW_QOS_CLASS_PCP_SET
50. GSW_QOS_DSCP_CLASS_GET
51. GSW_QOS_DSCP_CLASS_SET
52. GSW_QOS_DSCP_DROP_PRECEDENCE_CFG_GET
53. GSW_QOS_DSCP_DROP_PRECEDENCE_CFG_SET
54. GSW_QOS_METER_CFG_GET
55. GSW_QOS_METER_CFG_SET
56. GSW_QOS_METER_PORT_ASSIGN
57. GSW_QOS_METER_PORT_DEASSIGN
58. GSW_QOS_METER_PORT_GET
59. GSW_QOS_PCP_CLASS_GET
60. GSW_QOS_PCP_CLASS_SET
61. GSW_QOS_PORT_CFG_GET
62. GSW_QOS_PORT_CFG_SET
63. GSW_QOS_PORT_REMARKING_CFG_GET

64. GSW_QOS_PORT_REMARKING_CFG_SET
65. GSW_QOS_QUEUE_PORT_GET
66. GSW_QOS_QUEUE_PORT_SET
67. GSW_QOS_SCHEDULER_CFG_GET
68. GSW_QOS_SCHEDULER_CFG_SET
69. GSW_QOS_SHAPER_CFG_GET
70. GSW_QOS_SHAPER_CFG_SET
71. GSW_QOS_SHAPER_QUEUE_ASSIGN
72. GSW_QOS_SHAPER_QUEUE_DEASSIGN
73. GSW_QOS_SHAPER_QUEUE_GET
74. GSW_QOS_STORM_CFG_GET
75. GSW_QOS_STORM_CFG_SET
76. GSW_QOS_WRED_CFG_GET
77. GSW_QOS_WRED_CFG_SET
78. GSW_QOS_WRED_QUEUE_CFG_GET
79. GSW_QOS_WRED_QUEUE_CFG_SET
80. GSW_RMON_CLEAR
81. GSW_RMON_PORT_GET
82. GSW_STP_BPDU_RULE_GET
83. GSW_STP_BPDU_RULE_SET
84. GSW_STP_PORT_CFG_GET
85. GSW_STP_PORT_CFG_SET
86. GSW_VERSION_GET
87. GSW_VLAN_ID_CREATE
88. GSW_VLAN_ID_DELETE
89. GSW_VLAN_ID_GET
90. GSW_VLAN_PORT_CFG_GET
91. GSW_VLAN_PORT_CFG_SET
92. GSW_VLAN_PORT_MEMBER_ADD
93. GSW_VLAN_PORT_MEMBER_READ
94. GSW_VLAN_PORT_MEMBER_REMOVE
95. GSW_VLAN_MEMBER_INIT
96. GSW_VLAN_RESERVED_ADD
97. GSW_VLAN_RESERVED_REMOVE
98. GSW_WOL_CFG_GET
99. GSW_WOL_CFG_SET
100. GSW_WOL_PORT_CFG_GET
101. GSW_WOL_PORT_CFG_SET
102. GSW_IRQ_GET
103. GSW_IRQ_MASK_GET
104. GSW_IRQ_MASK_SET
105. GSW_IRQ_STATUS_CLEAR
106. GSW_PCE_RULE_DELETE
107. GSW_PCE_RULE_READ
108. GSW_PCE_RULE_WRITE
109. GSW_REGISTER_GET
110. GSW_REGISTER_SET
111. GSW_RESET
112. GSW_RMON_EXTEND_GET
113. GSW_TIMESTAMP_TIMER_GET
114. GSW_TIMESTAMP_TIMER_SET

115. GSW_TIMESTAMP_PORT_READ
116. GSW_TRUNKING_CFG_GET
117. GSW_TRUNKING_CFG_SET
118. GSW_TRUNKING_PORT_CFG_GET
119. GSW_TRUNKING_PORT_CFG_SET
120. GSW_QOS_WRED_PORT_CFG_GET
121. GSW_QOS_WRED_PORT_CFG_SET
122. GSW_QOS_FLOWCTRL_CFG_GET
123. GSW_QOS_FLOWCTRL_CFG_SET
124. GSW_QOS_FLOWCTRL_PORT_CFG_GET
125. GSW_QOS_FLOWCTRL_PORT_CFG_SET
126. GSW_QOS_QUEUE_BUFFER_RESERVE_CFG_GET
127. GSW_QOS_QUEUE_BUFFER_RESERVE_CFG_SET
128. GSW_SVLAN_CFG_GET
129. GSW_SVLAN_CFG_SET
130. GSW_SVLAN_PORT_CFG_GET
131. GSW_SVLAN_PORT_CFG_SET
132. GSW_QOS_SVLAN_CLASS_PCP_PORT_GET
133. GSW_QOS_SVLAN_CLASS_PCP_PORT_SET
134. GSW_QOS_SVLAN_PCP_CLASS_GET
135. GSW_QOS_SVLAN_PCP_CLASS_SET
136. GSW_PCE_EG_VLAN_CFG_SET
137. GSW_PCE_EG_VLAN_CFG_GET
138. GSW_PCE_EG_VLAN_ENTRY_WRITE
139. GSW_PCE_EG_VLAN_ENTRY_READ
140. GSW_PMAC_BM_CFG_GET
141. GSW_PMAC_EG_CFG_GET
142. GSW_PMAC_IG_CFG_GET
143. GSW_PMAC_COUNT_GET
144. GSW_RMON_MODE_SET
145. GSW_RMON_IF_GET
146. GSW_RMON_REDIRECT_GET
147. GSW_RMON_ROUTE_GET
148. GSW_RMON_METER_GET
149. GSW_QOS_METER_ACT
150. GSW_PMAC_GLBL_CFG_SET
151. GSW_PMAC_GLBL_CFG_GET
152. GSW_CTP_PORT_ASSIGNMENT_SET
153. GSW_CTP_PORT_ASSIGNMENT_GET
154. GSW_EXTENDEDVLAN_SET
155. GSW_EXTENDEDVLAN_GET
156. GSW_EXTENDEDVLAN_FREE
157. GSW_VLANFILTER_SET
158. GSW_VLANFILTER_GET
159. GSW_VLANFILTER_FREE
160. GSW_BRIDGE_CONFIG_SET
161. GSW_BRIDGE_CONFIG_GET
162. GSW_BRIDGE_FREE
163. GSW_CTP_PORT_CONFIG_SET
164. GSW_CTP_PORT_CONFIG_GET
165. GSW_CTP_PORT_CONFIG_RESET

166. GSW_BRIDGE_PORT_CONFIG_SET
167. GSW_BRIDGE_PORT_CONFIG_GET
168. GSW_BRIDGE_PORT_ALLOC
169. GSW_BRIDGE_PORT_FREE
170. GSW_EXTENDEDVLAN_ALLOC
171. GSW_VLANFILTER_ALLOC
172. GSW_BRIDGE_ALLOC
173. GSW_PMAC_RMON_GET
174. GSW_DEBUG_CTPTABLE_STATUS
175. GSW_DEBUG_BRDGPORTTABLE_STATUS
176. GSW_DEBUG_BRDGTABLE_STATUS
177. GSW_DEBUG_EXVLANTABLE_STATUS
178. GSW_DEBUG_VLANFILTERTABLE_STATUS
179. GSW_DEBUG_METERTABLE_STATUS
180. GSW_DEBUG_DSCP2PCPTABLE_STATUS
181. GSW_DEBUG_PMAPPER_STATUS
182. GSW_DEBUG_PMAC_EG
183. GSW_DEBUG_PMAC_IG
184. GSW_DEBUG_PMAC_BP
185. GSW_DEBUG_DEF_QMAP
186. GSW_DEBUG_DEF_BYP_QMAP
187. GSW_PMAC_BM_CFG_SET
188. GSW_PMAC_EG_CFG_SET
189. GSW_PMAC_IG_CFG_SET
190. GSW_DEBUG_CTP_STATISTICS
191. GSW_DEBUG_LP_STATISTICS
192. gswss
193. GSW_CTP_PORT_ASSIGNMENT_ALLOC
194. GSW_CTP_PORT_ASSIGNMENT_FREE
195. GSW_IRQ_REGISTER
196. GSW_IRQ_UNREGISTER
197. GSW_IRQ_ENABLE
198. GSW_IRQ_DISBALE
199. GSW_DEBUG_PRINT_PCEIRQ_LIST
200. GSW_RMON_FLOW_GET
201. GSW_RMON_TFLOW_CLEAR
202. GSW_DSCP2PCP_MAP_GET
203. GSW_DEFAUL_MAC_FILTER_GET
204. GSW_DEFAUL_MAC_FILTER_SET
205. GSW_QOS_COLOR_MARKING_TABLE_SET
206. GSW_QOS_COLOR_MARKING_TABLE_GET
207. GSW_QOS_COLOR_REMARKING_TABLE_GET
208. GSW_QOS_COLOR_REMARKING_TABLE_SET
209. GSW_DEBUG_RMON_PORT_GET
210. GSW_CTP_PORT_RMON_GET
211. GSW_BRIDGE_PORT_RMON_GET
212. GSW_CTP_PORT_PCEBYPASS_RMON_GET
213. GSW_DEBUG_RMON_ALL_GET
214. GSW_PBB_TUNNEL_TEMPLATE_ALLOC
215. GSW_PBB_TUNNEL_TEMPLATE_FREE
216. GSW_PBB_TUNNEL_TEMPLATE_SET

```
217. GSW_PBB_TUNNEL_TEMPLATE_GET
218. GSW_DEBUG_TUNNELTEMP_STATUS
219. GSW_LPID_TO_GPID_ASSIGNMENT_SET
220. GSW_LPID_TO_GPID_ASSIGNMENT_GET
221. GSW_GPID_TO_LPID_ASSIGNMENT_SET
222. GSW_GPID_TO_LPID_ASSIGNMENT_GET
223. GSW_DUMP_MEM
224. GSW_DEFAUL_PCE_QMAP_GET
225. GSW_DEFAUL_PCE_BYPASS_QMAP_GET
226. GSW_IG_CFG_PMAC_GET
227. GSW_DEBUG_PMAC_RMON_GET_ALL
228. GSW_PCE_RULE_ALLOC
229. GSW_PCE_RULE_FREE
230. GSW_DEBUG_PCERULETABLE_STATUS
231. GSW_VXLAN_CFG_SET
232. GSW_VXLAN_CFG_GET
233. GSW_READ_PCE_TABLE
234. GSW_QOS_METER_ALLOC
235. GSW_QOS_METER_FREE
236. GSW_PCE_RULE_ENABLE
237. GSW_PCE_RULE_DISABLE
238. GSW_VLAN_COUNTER_MAPPING_SET
239. GSW_VLAN_COUNTER_MAPPING_GET
240. GSW_VLAN_RMON_GET
241. GSW_VLAN_RMON_CONTROL_SET
242. GSW_VLAN_RMON_CONTROL_GET
243. GSW_VLAN_RMON_CLEAR
```

Note: This listing applies to 10G PON Chipset System Package 1.22.0.
The printout for other versions may vary.
Some of the commands are not supported by the PON hardware platforms, which include the gigabit switch version 0x31 (GSWIP 3.1).

14.3.8.1 Switch CLI Function Details

The command listing covers all the functions provided by the switch API, for all supported switch operating modes and hardware platforms. Some of the functions are only available in a specific operation mode and result in an error response when called in other modes. Some functions are restricted to certain versions of the GSWIP hardware. Only the functions for GSWIP-3.1 are supported by this software.

For information about the command parameters type, use:

```
# switch_cli <command_name>
```

Switch CLI Example 1

```
# switch_cli GSW_CTP_PORT_ASSIGNMENT_GET ?
GSW_CTP_PORT_ASSIGNMENT_GET
Command returns with error -2

Command Description:
--------------------
GSW_CTP_PORT_ASSIGNMENT_GET
        Supported Parameters:
        ---------------------
```

```
1. nLogicalPortId
```

Here a single command line parameter is required to be entered as `name=value`, for example:

```
# switch_cli GSW_CTP_PORT_ASSIGNMENT_GET nLogicalPortId=2
GSW_CTP_PORT_ASSIGNMENT_GET

          nLogicalPortId          = 2
          nFirstCtpPortId         = 10
          nNumberOfCtpPort        = 256
          eMode                   = 2
```

For more details about the switch CLI functions and the function parameters, refer to the in-line source code documentation found in `/ugw_sw/source/linux_prx300/include/net/switch_api` or the derived Doxygen documentation.

Switch CLI Example 2

Forward all packets with MAC destination address in the range of 01:80:C2:00:00:00 to 01:80:C2:00:00:FF to the CPU port:

```
switch_cli GSW_PCE_RULE_WRITE
pattern.nIndex=28
pattern.bMAC_DstEnable=1
pattern.nMAC_Dst=01:80:C2:00:00:00
pattern.nMAC_SrcMask=0x03
action.ePortMapAction=4
action.nForwardPortMap=0x1
```


14.3.8.2 Internal Numbering

The usage of the `switch_cli` functions requires the knowledge of the functional entities internal numbering to be accessed. These numbers are essential.

Table 88 Internal Numbering External Interfaces

Entity	Local Interface 1	Local Interface 2	WAN Interface	Remarks
SFU				
Port speed	Up to 10 Gbit/s, depending on external Ethernet PHY (XFI interface)	Up to 2.5 Gbit/s, using the internal GPHY	Up to 10 Gbit/s, depending on the optical interface and PON IP operation mode	The settings are fixed for a given application.
Port Name	LAN0 or UNI1	LAN1 or UNI2	PON	
Linux netdev Name	eth0_0	eth0_1	pon0	
Ethernet MAC	XGMAC1	XGMAC2	XGMAC0	This is a fixed assignment.
Logical MAC	MAC3	MAC4	MAC2	This is a fixed assignment.
Logical Port ID `nLogicalPortId`	3	4	2	This is a fixed assignment.
SFP				

Table 88 Internal Numbering External Interfaces (cont'd)

Entity	Local Interface 1	Local Interface 2	WAN Interface	Remarks
Port Speed	Not used.	Up to 10 Gbit/s, with the XFI interface or up to 2.5 Gbit/s with the SGMII interface	Up to 10 Gbit/s, depending on the optical interface and PON IP operation mode	The settings are fixed for a given application.
Port Name		LAN0	PON	
Linux netdev Name		eth0_0	pon0	
Ethernet MAC		XGMAC2	XGMAC0	This is a fixed assignment.
Logical MAC		MAC4	MAC2	This is a fixed assignment.
Logical Port ID `nLogicalPortId`		4	2	This is a fixed assignment.

Table 89 Internal Numbering of Internal Pseudo MAC Interfaces

Entity	Internal Port 0 Stage1	Internal Port 1 Stage1	Internal Port 0 Stage 2	Internal Port 1 Stage 2	Remark
Usage	Upstream from GSWIP to CQEM	Downstream from GSWIP to CQEM	Downstream from CQEM to GSWIP bypass	Upstream from CQEM to GSWIP bypass	The settings are fixed for a given application.
PMAC Name	PMAC0	PMAC1	PMAC0	PMAC1	This is a fixed assignment.
Port ID(s)	0, 3 to 11	2	0	1	This is a fixed assignment.
Direction	Egress	Egress	Ingress	Ingress	

Table 90 Internal Numbering of other Modules

Entity	Local Interface 1	Local Interface 2	WAN Interface	Remark
SFU				
Connection Termination Point (CTP)	`switch_cli GSW_CTP_PORT_ASSIGNMENT_GET nLogicalPortId=<n>` n = 0 to 11			Variable assignment, use the `nLogicalPortId` to identify. Multiple CTPs are assignable, numbered in ascending order.
	`switch_cli GSW_DEBUG_CTPTABLE_STATUS Checkall=1`			Shows all CTP in use and their related BP.
Sub-interface `nSubIfIdGroup`	`switch_cli GSW_CTP_PORT_CONFIG_GET nLogicalPortId=<n> | grep nSubIfIdGroup` n = 0 to 11			Variable assignment, use the `nLogicalPortId` to identify. A single sub-interface is assigned to a CTP.
Bridge Port (BP)	`switch_cli GSW_CTP_PORT_CONFIG_GET nLogicalPortId=<n> | grep BridgePort` n = 0 to 11			Variable assignment, use the `nLogicalPortId` to identify. A single BP is assigned to a CTP.
Bridge Number	`switch_cli GSW_BRIDGE_PORT_CONFIG_GET nBridgePortId=<n> | grep BridgeId` n = 0 to 127			Variable assignment, use the `nBridgePortId` to find the related `nBridgeId`.
Bridges in Use	`switch_cli GSW_DEBUG_BRDGTABLE_STATUS Checkall=1`			Shows all `nBridgeId` values in use.
QoS Priority Mapper (IEEE 802.1p-mapper)	`switch_cli GSW_BRIDGE_PORT_CONFIG_GET nBridgePortId=<n> | grep Pmapper` n = 0 to 127			Variable assignment, use the `nBridgePortId` to find the related `nPmapperId`.

14.3.9 Ethernet PHY Debug Access

The integrated and external Ethernet PHYs can be accessed through the internal and external MDIO interface. The register description of the integrated PHY can be taken from the data sheet of the standalone 2.5G PHY [17].

For a description of an external PHY's registers refer to the external PHY's manufacturer.

The MDIO interface can be accessed through the `switch_cli xgmac` debug functions.

The command structure is:

```
# switch_cli xgmac <index> <command> <argument1> <argument2> …
```

with

- <index>
  - This is the Ethernet MAC index, for the LAN interfaces use 3.
- <command>
  - `mdio_rd`
    Read a single MDIO register value.
  - `mdio_rd_cont`
    Read a range of MDIO register values.
  - `mdio_wr`
    Write a single MDIO register value.
- <argument1>
  - MDIO device address, these values are supported by the internal GPHY: 0, 1, 3, 7, 30, 31
    See [17] chapter 4.3 for more detail.
- <argument2>
  - PHY ID, the ID to identify the selected PHY on the MDIO bus.
    On the PRX reference designs the internal GPHY uses the value 31, the external PHY uses the value 0.
- <argument3>
  - `mdio_rd`: This is the PHY register address, in a range from 0 to 31.
  - `mdio_rd_cont`: This is the PHY register *start* address, in a range from 0 to 31.
- <argument4>
  - `mdio_rd_cont`: This is the PHY register *end* address, in a range from 0 to 31.
    The end address *must* be larger or equal to the start address!
  - `mdio_wr`: This is the data value to be written, 0x0000 to 0xFFFF.

The data values can be given in decimal or hexadecimal (0x…) format.

MDIO Example: Single Read from the Internal PHY

```
# switch_cli xgmac 3 mdio_rd 0 0x1F 0x00
xgmac
switch_cli xgmac 3 mdio_rd
OP      CL      DEVADR  PHYID   PHYREG  DATA
========================================
RD      CL45         0     1F        0  3040
```

MDIO Example: Continuous Read from the Internal PHY

```
# switch_cli xgmac 3 mdio_rd_cont 0 0x1F 0x00 0x3
xgmac
switch_cli xgmac 3 mdio_rd_cont
OP      CL      DEVADR  PHYID   PHYREG  DATA
=========================================
RD      CL45         0     1F        0  3040
RD      CL45         0     1F        1  796D
RD      CL45         0     1F        2  67C9
RD      CL45         0     1F        3  DC00
```

MDIO Example: Write to the Internal PHY

```
# switch_cli xgmac 3 mdio_wr 7 0x1f 0x00 0x3200
xgmac
```

Note: This example triggers auto-negotiation, you will see the link going down and up again.

14.3.10 Debug File System

A number of debug files are available at `/sys/kernel/debug`, when the image has been build with debug included. For productive software images, this must not be included for security reasons.

```
# ll /sys/kernel/debug
drwx------   26 root     root            0 Jan  1  1970 ./
drwxr-xr-x    7 root     root            0 Jan  1  1970 ../
drwxr-xr-x   20 root     root            0 Jan  1  1970 bdi/
drwxr-xr-x   70 root     root            0 Jan  1  1970 clk/
drwxr-xr-x    3 root     root            0 Jan  1  1970 cma/
drwxr-xr-x    2 root     root            0 Jan  1  1970 combophy/
drwxr-xr-x    2 root     root            0 Jan  1  1970 cqm/
drwxr-xr-x    2 root     root            0 Jan  1  1970 dma1rx/
drwxr-xr-x    2 root     root            0 Jan  1  1970 dma1tx/
drwxr-xr-x    2 root     root            0 Jan  1  1970 dma2rx/
drwxr-xr-x    2 root     root            0 Jan  1  1970 dma2tx/
drwxr-xr-x    2 root     root            0 Jan  1  1970 dp/
drwxr-xr-x    2 root     root            0 Jan  1  1970 dynamic_debug/
drwxr-xr-x    2 root     root            0 Jan  1  1970 extfrag/
-rw-r--r--    1 root     root            0 Jan  1  1970 fault_around_bytes
-r--r--r--    1 root     root            0 Jan  1  1970 gpio
drwxr-xr-x    2 root     root            0 Jan  1  1970 gptc/
-r--r--r--    1 root     root            0 Jan  1  1970 irq_domain_mapping
drwxr-xr-x    2 root     root            0 Jan  1  1970 mips/
drwxr-xr-x    3 root     root            0 Jan  1  1970 opp/
drwxr-xr-x    3 root     root            0 Jan  1  1970 pinctrl/
drwxr-xr-x    2 root     root            0 Jan  1  1970 pm_qos/
drwxr-xr-x    2 root     root            0 Jan  1  1970 pon_qos/
drwxr-xr-x    6 root     root            0 Jan  1  1970 pp/
drwxr-xr-x    4 root     root            0 Jan  1  1970 regmap/
drwxr-xr-x    2 root     root            0 Jan  1  1970 reset/
-r--r--r--    1 root     root            0 Jan  1  1970 sleep_time
drwxr-xr-x    2 root     root            0 Jan  1  1970 ssoled/
drwxr-xr-x    3 root     root            0 Jan  1  1970 ubi/
drwxr-xr-x    3 root     root            0 Jan  1  1970 ubifs/
```


14.3.11 GPIO Signals

To check the status of the GPIO signals, use:

```
# cat /sys/kernel/debug/gpio

gpiochip1: GPIOs 464-479, parent: platform/16320000.ssogpio, intel-sso-gpio:
 gpio-466 (                    |?                   ) out lo
 gpio-467 (                    |?                   ) out lo
 gpio-468 (                    |?                   ) out lo
 gpio-469 (                    |?                   ) out lo
 gpio-470 (                    |?                   ) out lo
 gpio-471 (                    |?                   ) out lo
 gpio-472 (                    |?                   ) out lo
 gpio-473 (                    |?                   ) out lo
 gpio-477 (                    |aqr_phy_reset       ) out hi
 gpio-479 (                    |?                   ) out lo
```

```
gpiochip0: GPIOs 480-511, parent: platform/16310000.pinctrl, gpiochip0:
 gpio-506 (                    |scl                 ) in  hi
 gpio-507 (                    |sda                 ) in  hi
```

To check the GPIO pin configuration, use:

```
# cat /sys/kernel/debug/pinctrl/16310000.pinctrl/pins
registered pins: 32
pin 0 (io-0) pin mux: 2
group name: qspi, mux: 2

pin 1 (io-1) pin mux: 2
group name: qspi, mux: 2

pin 2 (io-2) pin mux: 2
group name: qspi, mux: 2
group name: spi0_alt, mux: 3

pin 3 (io-3) pin mux: 2
group name: qspi, mux: 2
group name: spi0_alt, mux: 3

pin 4 (io-4) pin mux: 2
group name: qspi, mux: 2
group name: spi0_alt, mux: 3

pin 5 (io-5) pin mux: 2
group name: qspi, mux: 2
group name: spi0_cs_alt, mux: 3

pin 6 (io-6) pin mux: 2
group name: ssi, mux: 2
group name: spi0, mux: 3

pin 7 (io-7) pin mux: 2
group name: ssi, mux: 2
group name: spi0, mux: 3

pin 8 (io-8) pin mux: 2
group name: ssi_rx, mux: 2
group name: spi0, mux: 3

pin 9 (io-9) pin mux: 0
group name: ssi_rst_io, mux: 2
group name: tdm, mux: 4
group name: spi0_cs, mux: 3

pin 10 (io-10) pin mux: 1
group name: mdio0, mux: 1
group name: i2c1, mux: 2
group name: i2c1_slave, mux: 6
```

```
pin 11 (io-11) pin mux: 1
group name: mdio0, mux: 1
group name: i2c1, mux: 2
group name: i2c1_slave, mux: 6

pin 12 (io-12) pin mux: 3
group name: spi1, mux: 3

pin 13 (io-13) pin mux: 3
group name: spi1, mux: 3

pin 14 (io-14) pin mux: 2
group name: led, mux: 2

pin 15 (io-15) pin mux: 2
group name: led, mux: 2

pin 16 (io-16) pin mux: 2
group name: led, mux: 2

pin 17 (io-17) pin mux: 3
group name: spi1, mux: 3

pin 18 (io-18) pin mux: 0
group name: tdm, mux: 4

pin 19 (io-19) pin mux: 2
group name: ssi, mux: 2

pin 20 (io-20) pin mux: 2
group name: pon_ip, mux: 2

pin 21 (io-21) pin mux: 0
group name: pon_ip_sfp_tx_dis, mux: 2

pin 22 (io-22) pin mux: 2
group name: pon_ip, mux: 2

pin 23 (io-23) pin mux: 2
group name: pon_ip, mux: 2

pin 24 (io-24) pin mux: 2
group name: tdm, mux: 4
group name: mdio1, mux: 1
group name: pon_ip, mux: 2

pin 25 (io-25) pin mux: 0
group name: tdm, mux: 4
group name: mdio1, mux: 1
```

```
pin 26 (io-26) pin mux: 0
group name: i2c0, mux: 2
group name: i2c0_gpio, mux: 0

pin 27 (io-27) pin mux: 0
group name: i2c0, mux: 2
group name: i2c0_gpio, mux: 0

pin 28 (io-28) pin mux: 0
group name: gpc1, mux: 2

pin 29 (io-29) pin mux: 0
group name: gpc3, mux: 2

pin 30 (io-30) pin mux: 1
group name: uart1, mux: 1

pin 31 (io-31) pin mux: 1
group name: uart1, mux: 1
```

Note: This example has been taken from an EASYPRX321 reference design.

The individual GPIO signals are modified from the command line by using the standard Linux operations.

14.4 Data Path Debugging

The data packets go through three stages in the packer processing engine: GSWIP core, CQEM and GSWIP core PCE bypass. All stages provide counter information useful for debugging traffic problems.

14.4.1 GSWIP Core Traffic Flow Debugging

In the GSWIP core stage, there are counters per MAC, bridge port (BP) and connection termination point (CTP). The switch forwarding tables can also be read out. **Figure 150** shows the related probe points.

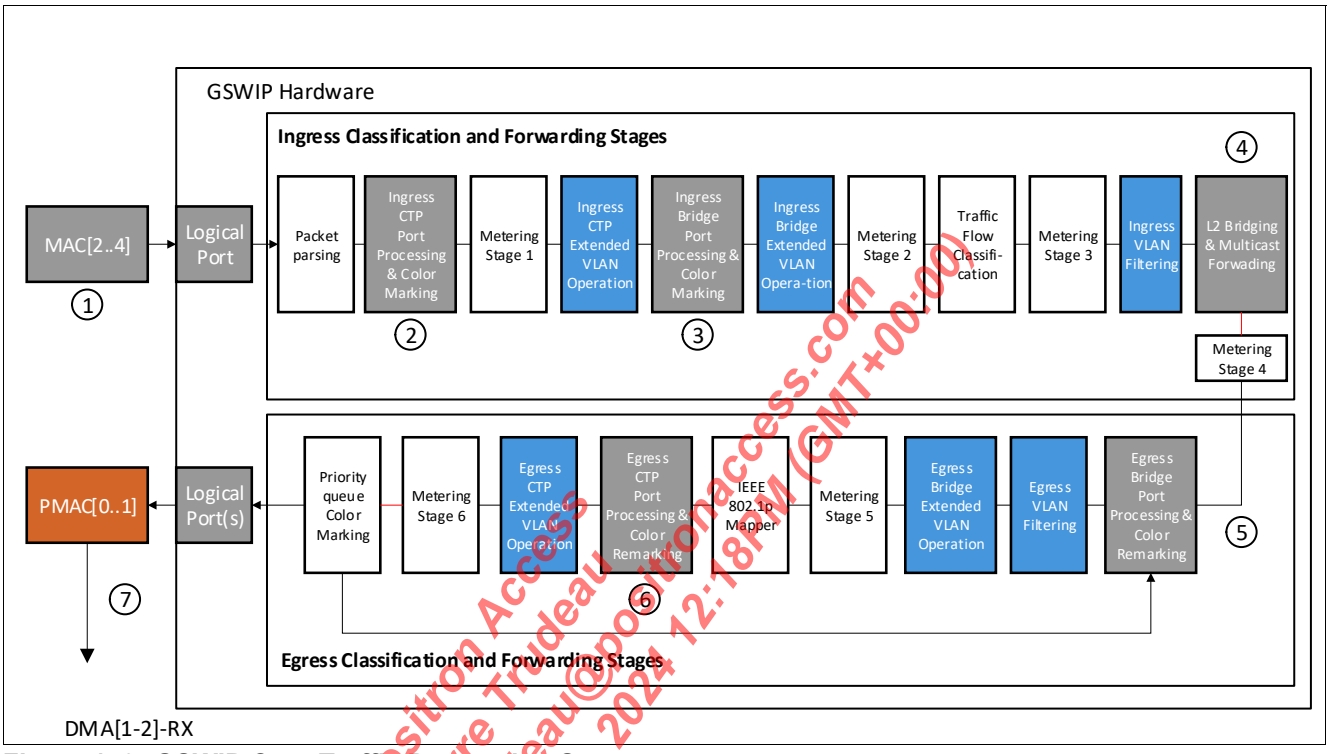


Figure 150 GSWIP Core Traffic Processing Stages

Table 91 Physical Ports MAC and Logical Port Map

	MAC	Logical Port
PON	0	2
LAN1	1	3
LAN2	2	4

Table 92 PMAC and Enqueue Port Assignments

	MAC	Logical Port
PON	0	2
LAN1	1	3
LAN2	2	4

Probe Point 1: MAC Counters

```
# switch_cli xgmac 2 get rmon
xgmac

TYPE                                XGMAC 2
Rx_Packets                =           37133
Rx_Bytes                  =        19012096
Rx_Byte_errors            =               0
Rx_Pauseframe             =               0
Rx_Crc_Errors             =               0
Rx_Fifo_Errors            =               0

Tx_Packets                =              34
Tx_Bytes                  =            8272
Tx_Packet_Errors          =               0
Tx_Byte_Errors            =               0
Tx_Pauseframe             =               0
Tx_underflow_error        =               0
```

To show all XGMAC counters together use

```
# switch_cli xgmac '*' get rmon
xgmac
XGMAC 2: Reading rmon
XGMAC 3: Reading rmon
XGMAC 4: Reading rmon

TYPE                                XGMAC 2             XGMAC 3             XGMAC 4
Rx_Packets                =                 0               47274                   0
Rx_Bytes                  =                 0             7255234                   0
Rx_Byte_errors            =                 0                   0                   0
Rx_Pauseframe             =                 0                   0                   0
Rx_Crc_Errors             =                 0                   0                   0
Rx_Fifo_Errors            =                 0                   0                   0

Tx_Packets                =                 0                  14                   0
Tx_Bytes                  =                 0                1732                   0
Tx_Packet_Errors          =                 0                   0                   0
Tx_Byte_Errors            =                 0                   0                   0
Tx_Pauseframe             =                 0                   0                   0
Tx_underflow_error        =                 0                   0                   0
```


Probe Points 2 and 6: Logical Port CTP Assignment

The matching CTP ID must be determined to be able to read its counters.

The logical port `nLogicalPortId==3` has four CTPs assigned and the first one on this logical port is `nPortID==2`. The other CTPs are numbered subsequently (3, 4, 5).

CTP Counters

The counters are separated by direction (ingress and egress).

Read the counters of a CTP with port ID=2 (the `nPortId` selector identifies the ID of the CTP port).

```
# switch_cli GSW_CTP_PORT_RMON_GET nPortId=2
GSW_CTP_PORT_RMON_GET

nRxGoodPkts                          : 37133
nRxUnicastPkts                       : 37133
nRxBroadcastPkts                     : 0
nRxMulticastPkts                     : 0
nRxFCSErrorPkts                      : 0
nRxUnderSizeGoodPkts                 : 0
nRxOversizeGoodPkts                  : 0
nRxUnderSizeErrorPkts                : 0
nRxOversizeErrorPkts                 : 0
nRxFilteredPkts                      : 5848
nRx64BytePkts                        : 0
nRx127BytePkts                       : 0
nRx255BytePkts                       : 0
nRx511BytePkts                       : 0
nRx1023BytePkts                      : 37133
nRxMaxBytePkts                       : 0
nRxDroppedPkts                       : 0
nRxExtendedVlanDiscardPkts           : 0
nMtuExceedDiscardPkts                : 0
nRxGoodBytes                         : 19012096 (0x1221a00)
nRxBadBytes                          : 0 (0x0)


nTxGoodPkts                          : 10
nTxUnicastPkts                       : 10
nTxBroadcastPkts                     : 0
nTxMulticastPkts                     : 0
nTx64BytePkts                        : 0
nTx127BytePkts                       : 0
nTx255BytePkts                       : 0
nTx511BytePkts                       : 0
nTx1023BytePkts                      : 10
nTxMaxBytePkts                       : 0
nTxDroppedPkts                       : 0
nTxOversizeGoodPkts                  : 0
nTxUnderSizeGoodPkts                 : 0
nTxAcmDroppedPkts                    : 0
nTxGoodBytes                         : 5120 (0x1400)
```

Probe Points 3 and 5: Bridge Port Counters

The counters are separated by direction (ingress and egress).

The bridge port for each CTP must be found prior to getting the according bridge port counters.

```
# switch_cli GSW_CTP_PORT_CONFIG_GET nLogicalPortId=3 nSubIfIdGroup=0
GSW_CTP_PORT_CONFIG_GET

          nLogicalPortId                  = 3
          nSubIfIdGroup                   = 0
          eMask                           = 0xffffffff
          nBridgePortId                   = 3
          bForcedTrafficClass             = 0
          nDefaultTrafficClass            = 0
          bIngressExtendedVlanEnable      = 0

               [...]
```

`nSubIfIdGroup` is the CTP(subif) index on the logical port. When the logical port 3 has four CTPs, call with `nSubIfIdGroup=[0…3]` to get the bridge port for each one of the previous command.

```
# switch_cli GSW_BRIDGE_PORT_RMON_GET nPortId=3
GSW_BRIDGE_PORT_RMON_GET

nRxGoodPkts                          : 37233
nRxUnicastPkts                       : 37233
nRxBroadcastPkts                     : 0
nRxMulticastPkts                     : 0
nRxFCSErrorPkts                      : 0
nRxUnderSizeGoodPkts                 : 0
nRxOversizeGoodPkts                  : 0
nRxUnderSizeErrorPkts                : 0
nRxOversizeErrorPkts                 : 0
nRxFilteredPkts                      : 13
nRx64BytePkts                        : 0
nRx127BytePkts                       : 0
nRx255BytePkts                       : 0
nRx511BytePkts                       : 0
nRx1023BytePkts                      : 37233
nRxMaxBytePkts                       : 0
nRxDroppedPkts                       : 0
nRxExtendedVlanDiscardPkts           : 5835
nMtuExceedDiscardPkts                : 0
nRxGoodBytes                         : 19063296 (0x122e200)
nRxBadBytes                          : 0 (0x0)


nTxGoodPkts                          : 10
nTxUnicastPkts                       : 10
nTxBroadcastPkts                     : 0
nTxMulticastPkts                     : 0
nTx64BytePkts                        : 0
nTx127BytePkts                       : 0
```

```
nTx255BytePkts                          : 0
nTx511BytePkts                          : 0
nTx1023BytePkts                         : 10
nTxMaxBytePkts                          : 0
nTxDroppedPkts                          : 0
nTxOversizeGoodPkts                     : 0
nTxUnderSizeGoodPkts                    : 0
nTxAcmDroppedPkts                       : 0
nTxGoodBytes                            : 5120 (0x1400)
```

Probe Point 4: Bridge ID and L2 Forwarding Table

To find the bridge ID (`nBridgeId`) for a given bridge port (`nBridgePortId`), the bridge port configuration must be checked.

```
# switch_cli GSW_BRIDGE_PORT_CONFIG_GET nBridgePortId=9
GSW_BRIDGE_PORT_CONFIG_GET

          nBridgePortId                                 = 0x9
          eMask                                         = 0xffffffff
          nBridgeId                                     = 0x2
          bIngressExtendedVlanEnable                    = 0x0
          nIngressExtendedVlanBlockId                   = 0x0
          bEgressExtendedVlanEnable                     = 0x0
          nEgressExtendedVlanBlockId                    = 0x0
          eIngressMarkingMode                           = 0x0
```

The bridge ID in this case equals 2.


```
# switch_cli GSW_MAC_TABLE_ENTRY_READ
GSW_MAC_TABLE_ENTRY_READ
-------------------------------------------------------------
   MAC Address     | Port  |  Age  |  SID  | FID  | Static
-------------------------------------------------------------
72:c9:94:ec:60:00 |    0  |    0  |    0  |2(key)|TRUE(static)
                                                 |nFilterFlag(key): 0x0
                                                 |PortMap      [0]: 0x1
...
                                                 |PortMap     [15]: 0x1
                                                 |Hit Status      : 0

00:50:f1:50:ee:23 |    0  |    0  |    0  |2(key)|TRUE(static)
                                                 |nFilterFlag(key): 0x0
                                                 |PortMap      [0]: 0x1
...
                                                 |PortMap     [15]: 0x1

00:50:f1:50:ee:25 |    0  |    0  |    0  |1(key)|TRUE(static)
                                                 |nFilterFlag(key): 0x0
-------------------------------------------------------------
```

Probe Point 7: PMAC Counters

The PMAC egress counters for traffic coming from GSWIP to CQEM is checked with this command. The `nPortID` in this case is the source logical port. The command checks the counters when the traffic is coming from MAC3:

```
# switch_cli GSW_PMAC_RMON_GET nPmacId=1 nPortId=3
GSW_PMAC_RMON_GET
          nPmacId                                 = 1
          nPortId                                 = 3
          Ingress Discard Packet Count            = 0
          Ingress Discard Byte Count              = 0
          Egress Checksum Error Packet Count      = 0
          Egress Checksum Error Byte Count        = 0
          Ingress Total Packet Count              = 0
          Ingress Total Byte Count                = 0
          Egress Total Packet Count               = 31685
          Egress Total Byte Count                 = 16349460
```

The next command checks the PMAC counters when the traffic is coming from PON MAC2:

```
#
switch_cli gsw_pmac_rmon_get npmacid=0 nportid=2

# switch_cli GSW_PMAC_RMON_GET nPmacId=1 nPortId=0
GSW_PMAC_RMON_GET
          nPmacId                                 = 1
          nPortId                                 = 0
          Ingress Discard Packet Count            = 0
          Ingress Discard Byte Count              = 0
          Egress Checksum Error Packet Count      = 0
          Egress Checksum Error Byte Count        = 0
          Ingress Total Packet Count              = 32293
          Ingress Total Byte Count                = 16529665
          Egress Total Packet Count               = 0
          Egress Total Byte Count                 = 0
```


Data Path Driver debugfs – CTP, Bridge Ports, and Network Devices

The port file shows all logical ports and the related configuration details.

```
# cat /sys/kernel/debug/dp/port

Reserved Port: rx_err_drop=0x00000000  tx_err_drop=0x00000000
          : bp=0(member:2 3 4 5 6 7 8 9  Fid=0 )
          : qid/node:    1/0
          : port/node:    0/0

01: rx_err_drop=0x00000000  tx_err_drop=0x00000000

02:module=0x0xc06a27a0(name:mod_pon_eth) dev_port=02 dp_port=02
itf_base=10(Enabled) ctp_max=256
    status:            PORT_SUBIF_REGISTERED
    allocate_flags:    GPON
    mode:              1
    LCT:               0
```

```
cb->rx_fn:          0xc069f24c
cb->restart_fn:     0x0
cb->stop_fn:        0x0
cb->get_subifid_fn:0x0
num_subif:          10
vap_offset/mask:    0/0xff
flag_other:         0xef
deq_port_base:      26
deq_port_num:       64
dma_chan:           0x3000000
tx_pkt_credit:      8
tx_b_credit:        00
tx_ring_addr:       0x241a00
tx_ring_size:       1
tx_ring_offset:     256(to next dequeue port)

  [00]:subif=0x0000 netif=0x61d0c800(netif=gem-omci),device_name=gem-omci
       : subif_flag = NULL
       : rx_fn_rxif_pkt =0x000001eb
       : rx_fn_txif_pkt =0x00000000
       : rx_fn_dropped  =0x00000000
       : tx_cbm_pkt     =0x000001f3
       : tx_tso_pkt     =0x00000000
       : tx_pkt_dropped =0x00000000
       : tx_clone_pkt   =0x00000000
       : tx_hdr_room_pkt=0x00000000
       : bp=7(member:0  Fid=1 )
       : qid/node:    5/0
       : port/node:    26/0(ref=2)
       : ctp_dev = NULL

  [01]:subif=0x0001 netif=0x6261f000(netif=gem65534),device_name=gem65534
       : subif_flag = NULL
       : rx_fn_rxif_pkt =0x00000000
       : rx_fn_txif_pkt =0x00000000
       : rx_fn_dropped  =0x00000000
       : tx_cbm_pkt     =0x00000000
       : tx_tso_pkt     =0x00000000
       : tx_pkt_dropped =0x00000000
       : tx_clone_pkt   =0x00000000
       : tx_hdr_room_pkt=0x00000000
       : bp=8(member:0  Fid= )
       : qid/node:    5/0
       : port/node:    26/0(ref=2)
       : ctp_dev = NULL

  [02]:subif=0x0002 netif=0x61df9000(netif=pmapper4354),device_name=pmapper4354
       : subif_flag = NULL
       : rx_fn_rxif_pkt =0x00000000
       : rx_fn_txif_pkt =0x00000000
       : rx_fn_dropped  =0x00000000
```

```
     : tx_cbm_pkt      =0x00000009
     : tx_tso_pkt      =0x00000000
     : tx_pkt_dropped =0x00000000
     : tx_clone_pkt    =0x00000000
     : tx_hdr_room_pkt=0x00000000
     : bp=9(member:3  Fid=2 )
     : qid/node:     6/0
     : port/node:     27/0(ref=8)
     : ctp_dev = gem1031

[03]:subif=0x0003 netif=0x61df9000(netif=pmapper4354),device_name=pmapper4354
     : subif_flag = NULL
     : rx_fn_rxif_pkt =0x00000000
     : rx_fn_txif_pkt =0x00000000
     : rx_fn_dropped  =0x00000000
     : tx_cbm_pkt      =0x00000000
     : tx_tso_pkt      =0x00000000
     : tx_pkt_dropped =0x00000000
     : tx_clone_pkt    =0x00000000
     : tx_hdr_room_pkt=0x00000000
     : bp=9(member:3  Fid=2 )
     : qid/node:     6/0
     : port/node:     27/0(ref=8)
     : ctp_dev = gem1055

[04]:subif=0x0004 netif=0x61df9000(netif=pmapper4354),device_name=pmapper4354
     : subif_flag = NULL
     : rx_fn_rxif_pkt =0x00000000
     : rx_fn_txif_pkt =0x00000000
     : rx_fn_dropped  =0x00000000
     : tx_cbm_pkt      =0x00000000
     : tx_tso_pkt      =0x00000000
     : tx_pkt_dropped =0x00000000
     : tx_clone_pkt    =0x00000000
     : tx_hdr_room_pkt=0x00000000
     : bp=9(member:3  Fid=2 )
     : qid/node:     6/0
     : port/node:     27/0(ref=8)
     : ctp_dev = gem1058

[05]:subif=0x0005 netif=0x61df9000(netif=pmapper4354),device_name=pmapper4354
     : subif_flag = NULL
     : rx_fn_rxif_pkt =0x00000000
     : rx_fn_txif_pkt =0x00000000
     : rx_fn_dropped  =0x00000000
     : tx_cbm_pkt      =0x00000000
     : tx_tso_pkt      =0x00000000
     : tx_pkt_dropped =0x00000000
     : tx_clone_pkt    =0x00000000
     : tx_hdr_room_pkt=0x00000000
     : bp=9(member:3  Fid=2 )
```

```
     : qid/node:    6/0
     : port/node:    27/0(ref=8)
     : ctp_dev = gem1064

[06]:subif=0x0006 netif=0x61df9000(netif=pmapper4354),device_name=pmapper4354
     : subif_flag = NULL
     : rx_fn_rxif_pkt =0x00000000
     : rx_fn_txif_pkt =0x00000000
     : rx_fn_dropped  =0x00000000
     : tx_cbm_pkt     =0x00000000
     : tx_tso_pkt     =0x00000000
     : tx_pkt_dropped =0x00000000
     : tx_clone_pkt   =0x00000000
     : tx_hdr_room_pkt=0x00000000
     : bp=9(member:3  Fid=2 )
     : qid/node:    6/0
     : port/node:    27/0(ref=8)
     : ctp_dev = gem1065

[07]:subif=0x0007 netif=0x61df9000(netif=pmapper4354),device_name=pmapper4354
     : subif_flag = NULL
     : rx_fn_rxif_pkt =0x00000000
     : rx_fn_txif_pkt =0x00000000
     : rx_fn_dropped  =0x00000000
     : tx_cbm_pkt     =0x00000000
     : tx_tso_pkt     =0x00000000
     : tx_pkt_dropped =0x00000000
     : tx_clone_pkt   =0x00000000
     : tx_hdr_room_pkt=0x00000000
     : bp=9(member:3  Fid=2 )
     : qid/node:    6/0
     : port/node:    27/0(ref=8)
     : ctp_dev = gem1066

[08]:subif=0x0008 netif=0x61df9000(netif=pmapper4354),device_name=pmapper4354
     : subif_flag = NULL
     : rx_fn_rxif_pkt =0x00000000
     : rx_fn_txif_pkt =0x00000000
     : rx_fn_dropped  =0x00000000
     : tx_cbm_pkt     =0x00000000
     : tx_tso_pkt     =0x00000000
     : tx_pkt_dropped =0x00000000
     : tx_clone_pkt   =0x00000000
     : tx_hdr_room_pkt=0x00000000
     : bp=9(member:3  Fid=2 )
     : qid/node:    6/0
     : port/node:    27/0(ref=8)
     : ctp_dev = gem1067

[09]:subif=0x0009 netif=0x61df9000(netif=pmapper4354),device_name=pmapper4354
     : subif_flag = NULL
```

```
          : rx_fn_rxif_pkt =0x00000000
          : rx_fn_txif_pkt =0x00000000
          : rx_fn_dropped  =0x00000000
          : tx_cbm_pkt     =0x00000000
          : tx_tso_pkt     =0x00000000
          : tx_pkt_dropped =0x00000000
          : tx_clone_pkt   =0x00000000
          : tx_hdr_room_pkt=0x00000000
          : bp=9(member:3  Fid=2 )
          : qid/node:    6/0
          : port/node:    27/0(ref=8)
          : ctp_dev = gem1068
    rx_err_drop=0x00000000  tx_err_drop=0x00000000

03:module=0x0xc0666d60(name:module00) dev_port=03 dp_port=03 itf_base=2(Enabled)
ctp_max=4
    status:             PORT_SUBIF_REGISTERED
    allocate_flags:     ETH_LAN
    mode:               2
    LCT:                1
    cb->rx_fn:          0xc0662a24
    cb->restart_fn:     0xc0661048
    cb->stop_fn:        0xc0663c48
    cb->get_subifid_fn:0x0
    num_subif:          2
    vap_offset/mask:    8/0xf
    flag_other:         0xef
    deq_port_base:      7
    deq_port_num:       1
    dma_chan:           0x1000000
    tx_pkt_credit:      8
    tx_b_credit:        00
    tx_ring_addr:       0x240700
    tx_ring_size:       1
    tx_ring_offset:     256(to next dequeue port)

      [00]:subif=0x0000 netif=0x62eb1000(netif=eth0_0),device_name=eth0_0
          : subif_flag = NULL
          : rx_fn_rxif_pkt =0x00000000
          : rx_fn_txif_pkt =0x00000000
          : rx_fn_dropped  =0x00000000
          : tx_cbm_pkt     =0x00000018
          : tx_tso_pkt     =0x00000000
          : tx_pkt_dropped =0x00000000
          : tx_clone_pkt   =0x00000000
          : tx_hdr_room_pkt=0x00000000
          : bp=3(member:9  Fid=2 )
          : qid/node:    2/0
          : port/node:    7/0(ref=2)
          : ctp_dev = NULL
```

```
      [01]:subif=0x0100
netif=0x62ebd000(netif=eth0_0_1_lct),device_name=eth0_0_1_lct
          : subif_flag = NULL
          : rx_fn_rxif_pkt =0x00000000
          : rx_fn_txif_pkt =0x00000000
          : rx_fn_dropped  =0x00000000
          : tx_cbm_pkt     =0x00000000
          : tx_tso_pkt     =0x00000000
          : tx_pkt_dropped =0x00000000
          : tx_clone_pkt   =0x00000000
          : tx_hdr_room_pkt=0x00000000
          : bp=4(member:0  Fid=0 )
          : qid/node:    2/0
          : port/node:    7/0(ref=2)
          : ctp_dev = NULL
    rx_err_drop=0x00000000  tx_err_drop=0x00000000

04:module=0x0xc0666ec0(name:module01) dev_port=04 dp_port=04 itf_base=6(Enabled)
ctp_max=4
    status:             PORT_SUBIF_REGISTERED
    allocate_flags:     ETH_LAN
    mode:               2
    LCT:                1
    cb->rx_fn:          0xc0662a24
    cb->restart_fn:     0xc0661048
    cb->stop_fn:        0xc0663c48
    cb->get_subifid_fn:0x0
    num_subif:          2
    vap_offset/mask:    8/0xf
    flag_other:         0xef
    deq_port_base:      15
    deq_port_num:       1
    dma_chan:           0x1000008
    tx_pkt_credit:      8
    tx_b_credit:        00
    tx_ring_addr:       0x240f00
    tx_ring_size:       1
    tx_ring_offset:     256(to next dequeue port)

      [00]:subif=0x0000 netif=0x62fd1000(netif=eth0_1),device_name=eth0_1
          : subif_flag = NULL
          : rx_fn_rxif_pkt =0x00000000
          : rx_fn_txif_pkt =0x00000000
          : rx_fn_dropped  =0x00000000
          : tx_cbm_pkt     =0x00000000
          : tx_tso_pkt     =0x00000000
          : tx_pkt_dropped =0x00000000
          : tx_clone_pkt   =0x00000000
          : tx_hdr_room_pkt=0x00000000
          : bp=5(member:0  Fid=0 )
          : qid/node:    3/0
```

```
          : port/node:    15/0(ref=2)
          : ctp_dev = NULL

      [01]:subif=0x0100
netif=0x62fd7000(netif=eth0_1_1_lct),device_name=eth0_1_1_lct
          : subif_flag = NULL
          : rx_fn_rxif_pkt =0x00000000
          : rx_fn_txif_pkt =0x00000000
          : rx_fn_dropped  =0x00000000
          : tx_cbm_pkt     =0x00000000
          : tx_tso_pkt     =0x00000000
          : tx_pkt_dropped =0x00000000
          : tx_clone_pkt   =0x00000000
          : tx_hdr_room_pkt=0x00000000
          : bp=6(member:0  Fid=0 )
          : qid/node:    3/0
          : port/node:    15/0(ref=2)
          : ctp_dev = NULL
    rx_err_drop=0x00000000  tx_err_drop=0x00000000
05: rx_err_drop=0x00000000  tx_err_drop=0x00000000
06: rx_err_drop=0x00000000  tx_err_drop=0x00000000
07: rx_err_drop=0x00000000  tx_err_drop=0x00000000
08: rx_err_drop=0x00000000  tx_err_drop=0x00000000
09: rx_err_drop=0x00000000  tx_err_drop=0x00000000
10: rx_err_drop=0x00000000  tx_err_drop=0x00000000
11: rx_err_drop=0x00000000  tx_err_drop=0x00000000
12: rx_err_drop=0x00000000  tx_err_drop=0x00000000
13: rx_err_drop=0x00000000  tx_err_drop=0x00000000
14: rx_err_drop=0x00000000  tx_err_drop=0x00000000
15: rx_err_drop=0x00000000  tx_err_drop=0x00000000
```

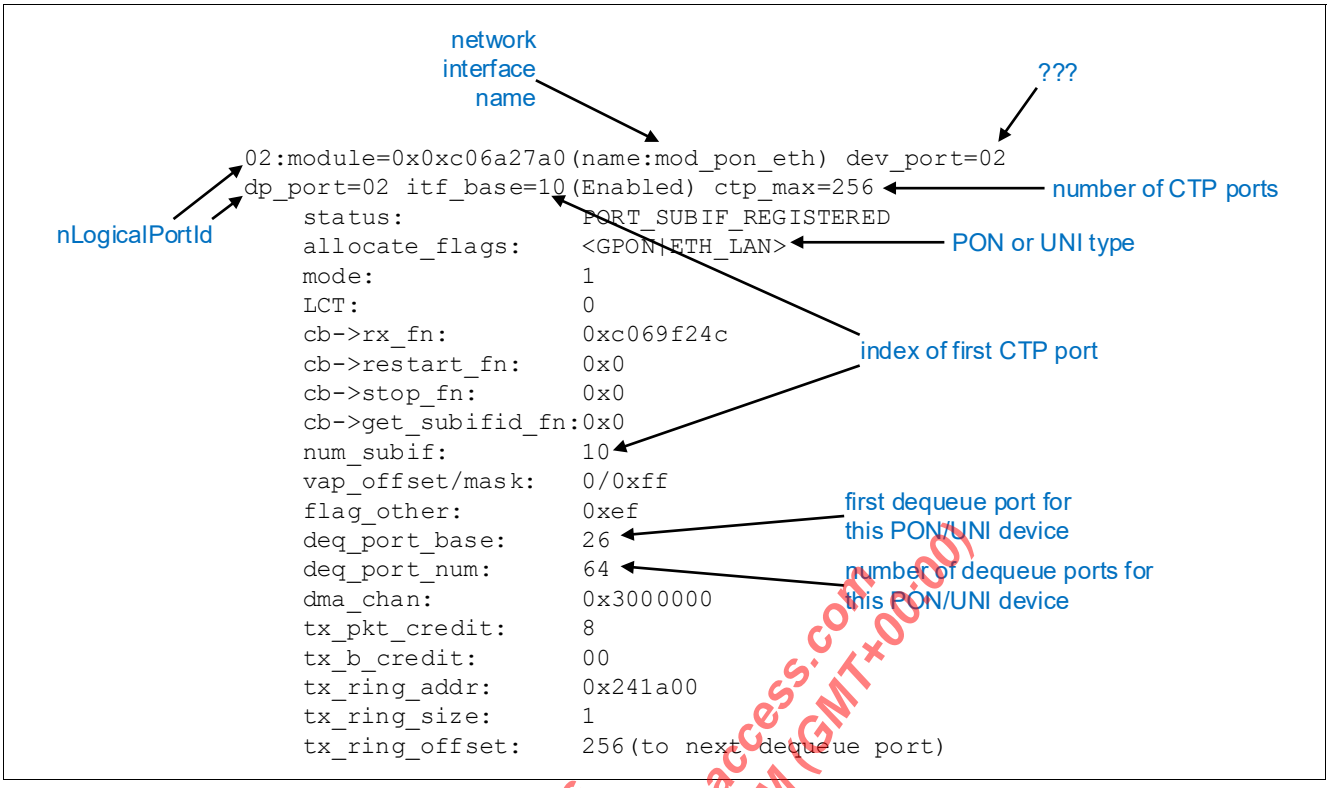


Figure 151 Data Path Port Internal Numbering Decoding – Part 1

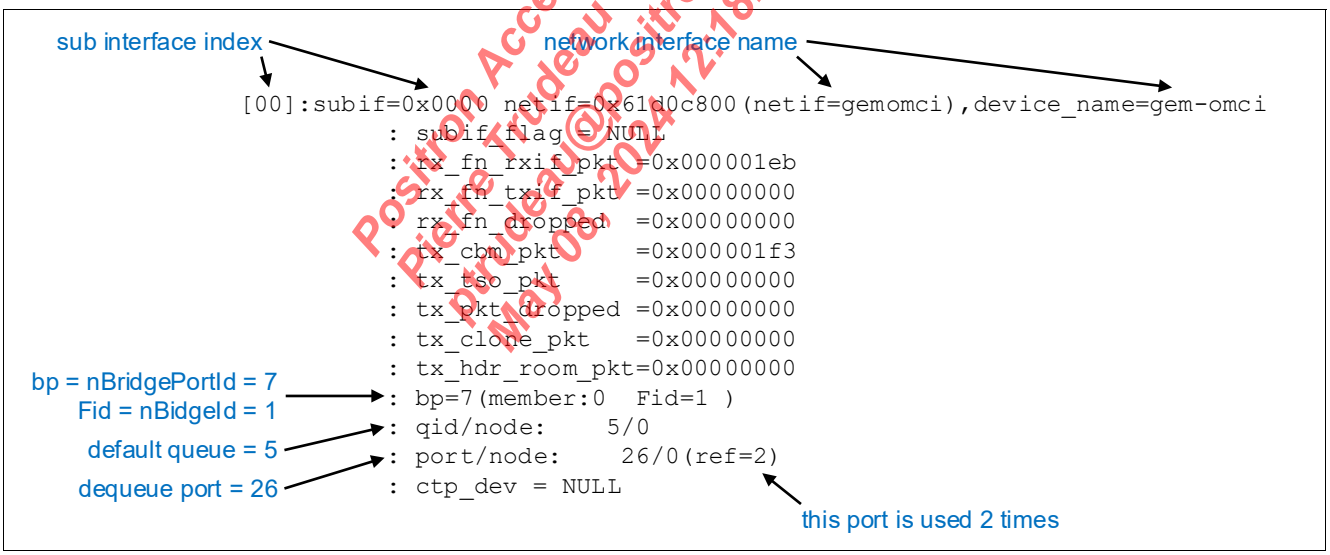


Figure 152 Data Path Port Internal Numbering Decoding – Part 2

14.4.2 CQEM Traffic Flow Debugging

These steps are required to analyze details of the traffic flow handling for the QoS in the CQEM hardware module.

14.4.2.1 CQEM Internal Numbering

Use this command to show the internal numbering used by the CQEM according to the applied configuration.

This is a first simple example with no PON configuration and fiber disconnected.

```
/# cat /sys/kernel/debug/dp/qos
Note:
  x/y :physical node/logical node
      :cqm dequeue port/ppv4 logical node
  Blk :blocked, ie new incoming packet will be dropped
  Susp:suspended, ie, packet in queue will not be scheduled
  C   :CIR in kbps
  P   :PIR in kbps
  c   :cbs in bytes
  p   :pbs in bytes
                                          ------------------
                                          |port[0/2]       |
                                          | stat:Normal    |
                   --------------------   |                |
                   |sched[/18]        |   |                |
                   | stat:Normal      |   |                |
q[   5/  20]------ |child[0]:WSP:0    |   |                |
  stat:Normal      | C/P:NoLim/     0 |   |                |
                   | c/p:16384/20480  |   |                |
q[   6/  21]------ |child[1]:WSP:1    |   |                |
  stat:Normal      | C/P: 8000/10000  |   |                |
                   | c/p: 8192/16384  |   |                |
q[   7/  22]------ |child[2]:WSP:2    |   |                |
  stat:Normal      | C/P: 8000/10000  |   |                |
                   | c/p: 8192/16384  |   |                |
q[   8/  23]------ |child[3]:WSP:3    |   |                |
  stat:Normal      | C/P: 8000/10000  |   |                |--CPU:0
                   | c/p: 8192/16384  |---|child[0] WSP:0  |  -
q[   9/  24]------ |child[4]:WSP:4    |   | C/P:NoLim/     0 |  C/P:NoLim/     0
  stat:Normal      | C/P: 8000/10000  |   | c/p:16384/     0 |  c/p:16384/     0
                   | c/p: 8192/16384  |   |                |
q[  10/  25]------ |child[5]:WSP:5    |   |                |
  stat:Normal      | C/P: 8000/10000  |   |                |
                   | c/p: 8192/16384  |   |                |
q[  11/  26]------ |child[6]:WSP:6    |   |                |
  stat:Normal      | C/P: 8000/10000  |   |                |
                   | c/p: 8192/16384  |   |                |
q[  12/  27]------ |child[7]:WSP:7    |   |                |
  stat:Normal      | C/P: 8000/10000  |   |                |
                   | c/p: 8192/16384  |   |                |
                   --------------------   |                |
q[   1/   5]-----------------------------|child[1]:WSP:7  |
  stat:Normal                            | C/P:NoLim/     0 |
                                         | c/p:16384/20480  |
```

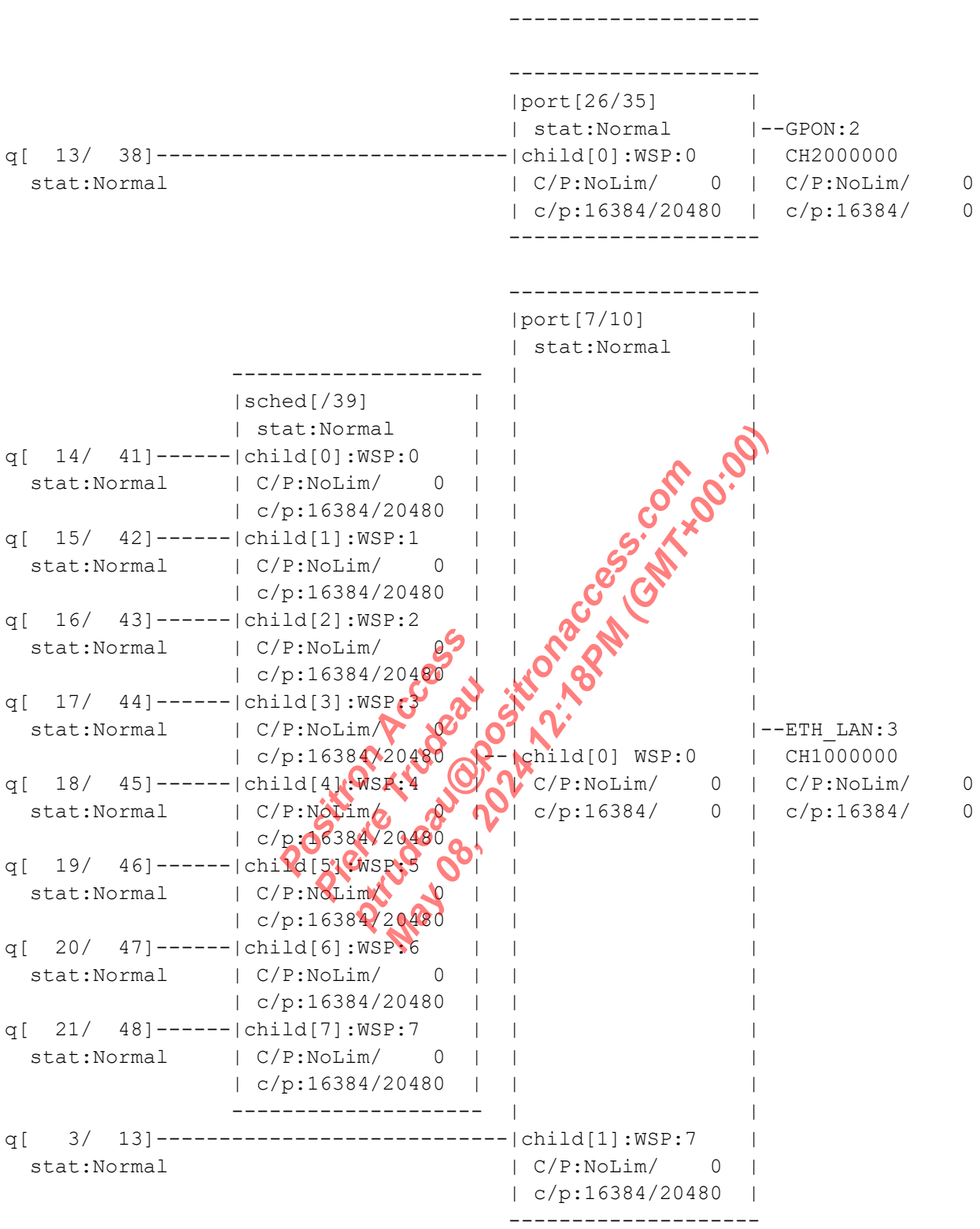


Figure 153 shows how the numbering is extracted.

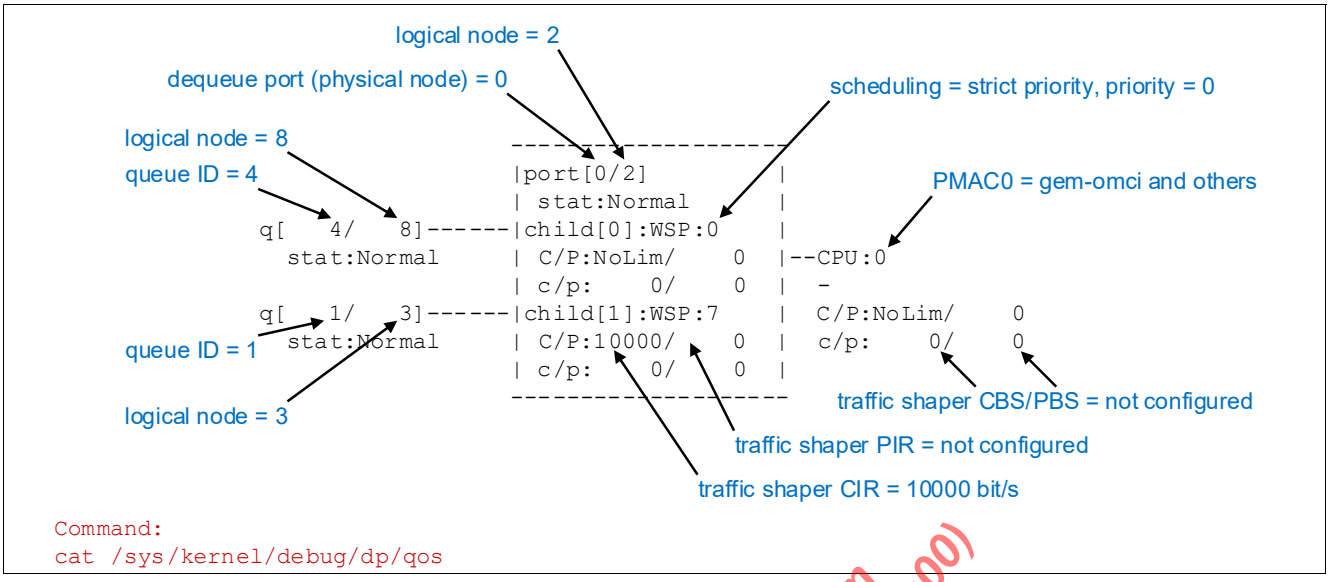


Figure 153 CQEM Internal Numbering Decoding

14.4.2.1.1 CQEM Queue Numbering

The queue assignment is analyzed through this command. **Figure 154** to **Figure 156** show the interpretation of this status table. The meaning of the fields depends on the indicated mode, which is:

- 0: CPU (see **Figure 154**)
- 2: PON port (see **Figure 155**)
- 3 or 4: UNI port (see **Figure 156**)

Other modes are not used in PON applications.

Figure 158 depicts the CQEM internal hardware structure.
Figure 157 shows the detailed queue encoding scheme.

```
# cat /sys/kernel/debug/dp/lookup

EP0 :   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=0
        x    x    x    x    x    x    0    0    0    0    x    x    x    0  ->1  (0x0000)
        x    x    x    x    x    x    0    0    0    0    x    x    0    1  ->1  (0x0001)
        x    x    x    x    x    x    0    0    0    0    x    0    1    1  ->1  (0x0003)
        x    x    x    x    x    x    0    0    0    0    0    1    1    1  ->4  (0x0007)
        x    x    x    x    x    x    0    0    0    0    1    1    1    1  ->1  (0x000f)
EP1 :   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=0
EP2 :   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=1
        0    0    0    0    x    x    0    0    1    0    0    x    1    x  ->6  (0x0022)
        0    0    0    0    x    x    0    0    1    0    0    0    0    x  ->5  (0x0020)
        0    0    0    0    x    x    0    0    1    0    1    0    0    x  ->6  (0x0028)
        0    0    0    0    x    x    0    0    1    0    0    1    0    x  ->6  (0x0024)
EP3 :   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=2
        0    0    0    x    x    x    0    0    1    1    x    x    x    x  ->2  (0x0030)
EP4 :   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=2
        0    0    0    x    x    x    0    1    0    0    x    x    x    x  ->3  (0x0040)
EP5 :   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=0
EP6 :   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=0
EP7 :   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=0
EP8 :   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=0
EP9 :   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=0
EP10:   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=0
EP11:   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=0
EP12:   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=0
EP13:   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=0
EP14:   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=0
EP15:   F2   F1  DEC  ENC MPE2 MPE1  EP3  EP2  EP1  EP0   C3   C2   C1   C0  qid   id mode=0
```

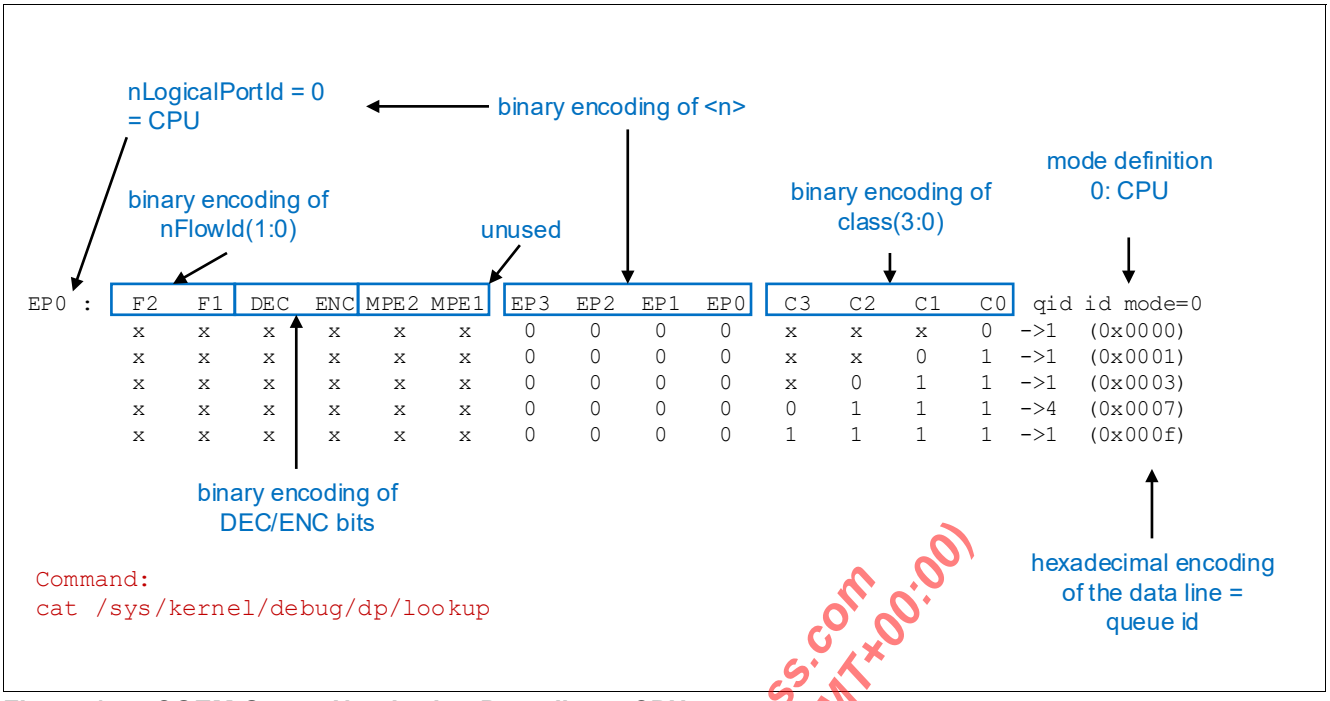


Figure 154 CQEM Queue Numbering Decoding – CPU

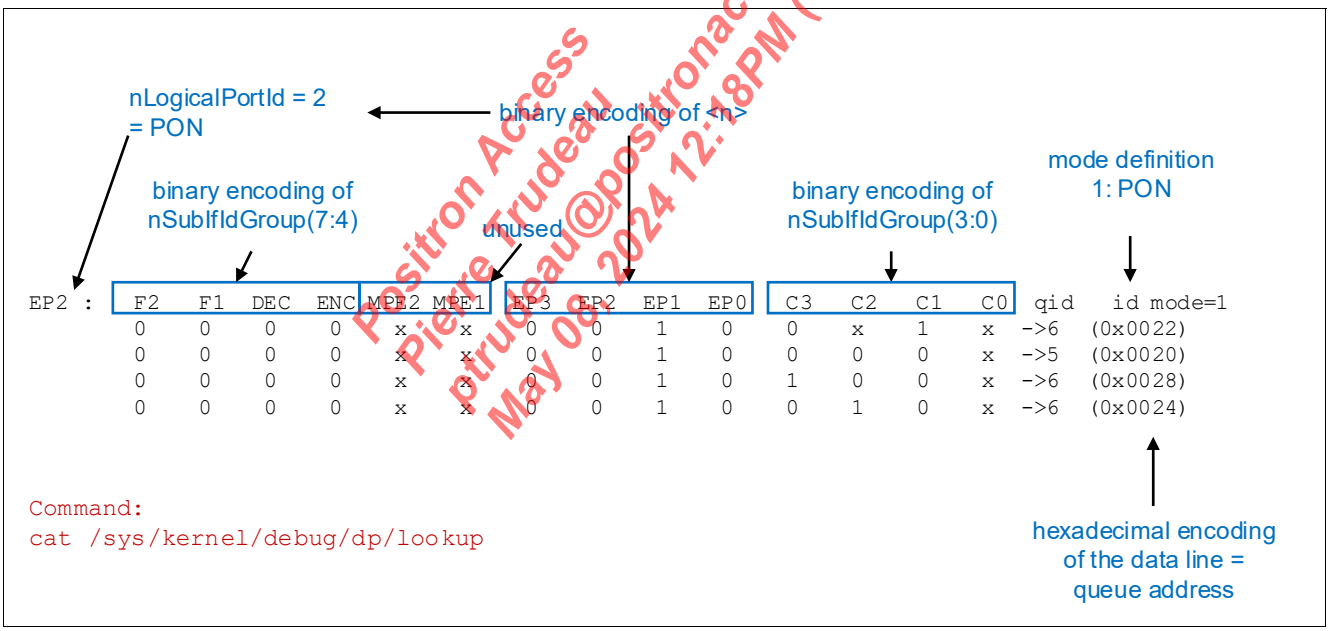


Figure 155 CQEM Queue Numbering Decoding – PON

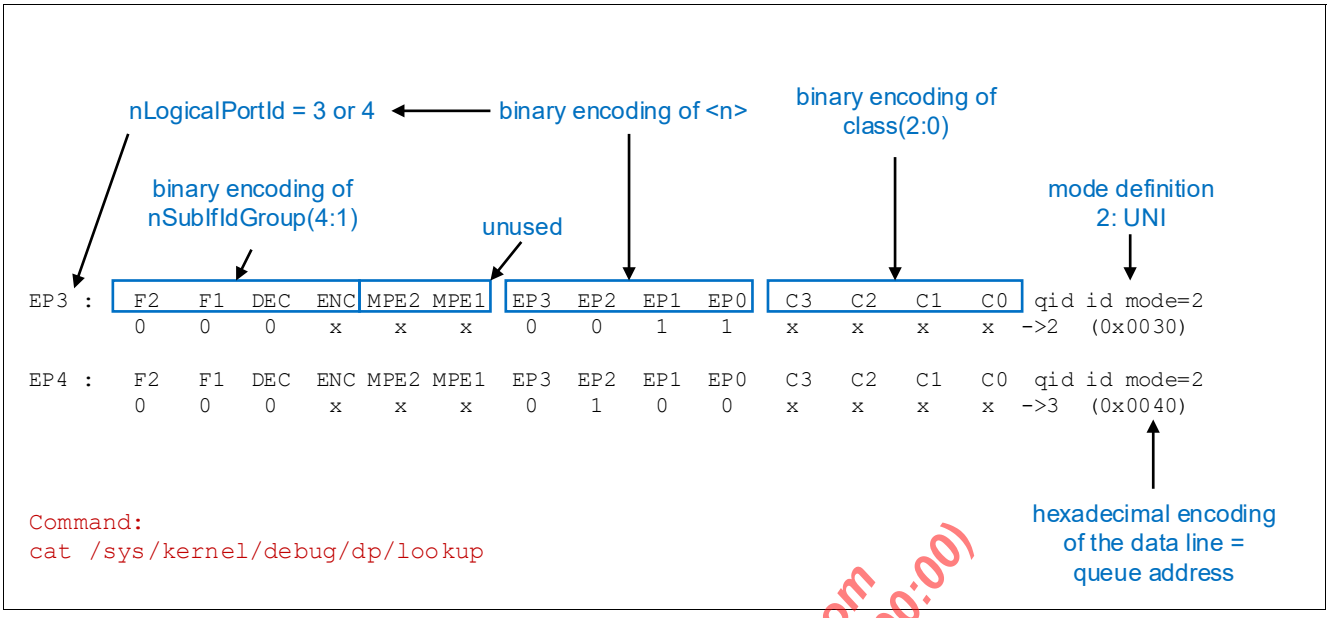


Figure 156 CQEM Queue Numbering Decoding – UNI

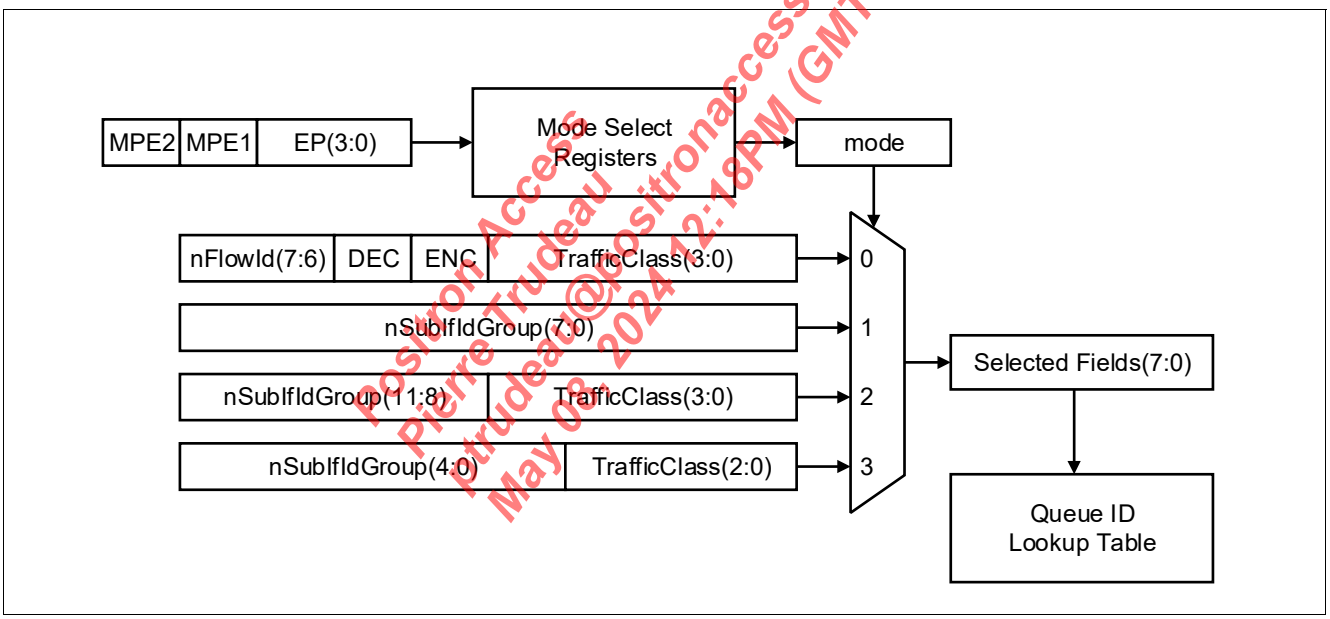


Figure 157 CQEM Queue Numbering

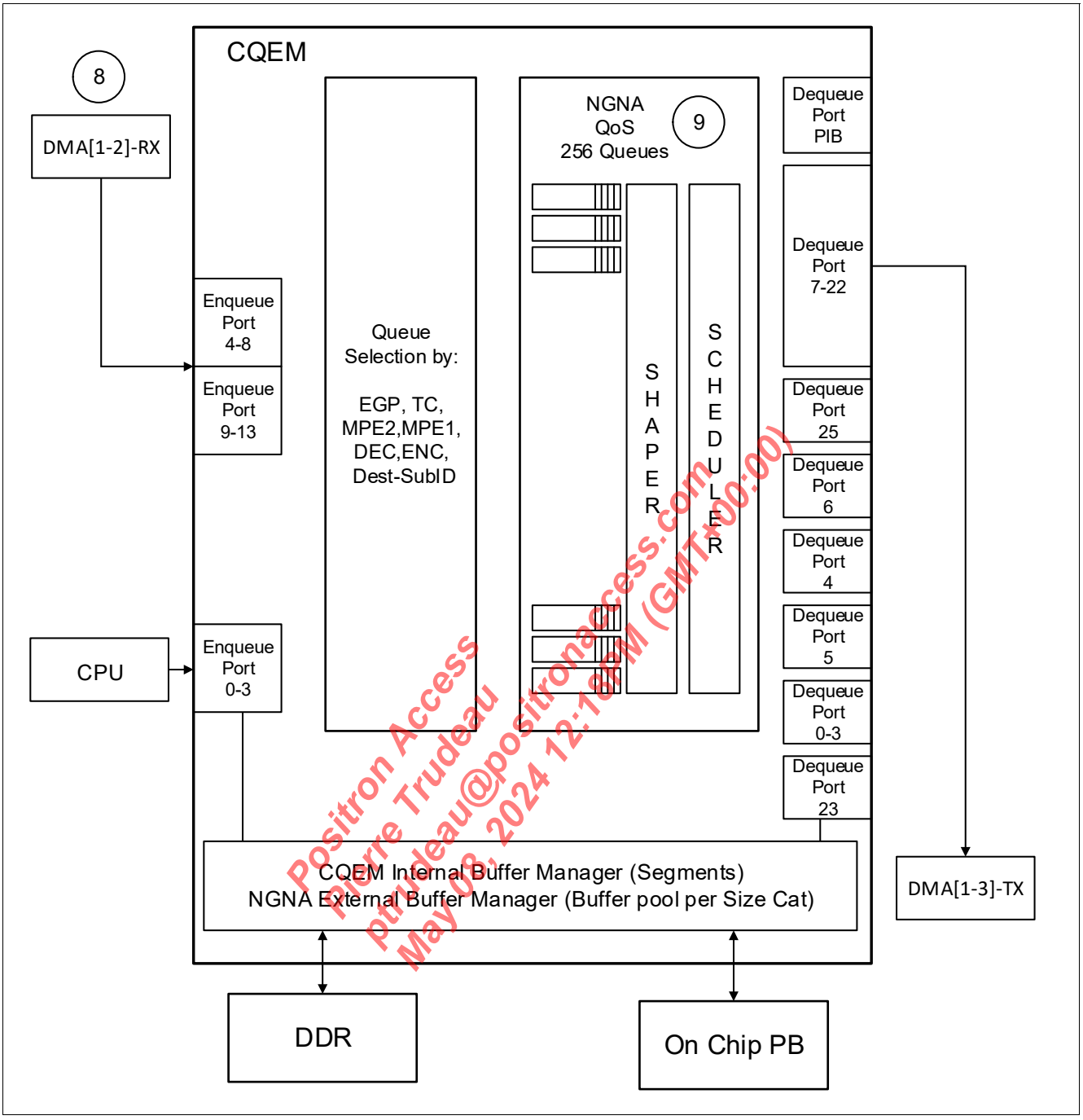


Figure 158 CQEM Hardware Block Diagram

14.4.2.2 DMA Counters

The internal DMA counters indicate whether packets are handled or do not reach the specific DMA hardware.

Probe Point 8: DMA Counters

```
# cat /sys/kernel/debug/cqm/cqmeqpc
Name: EQPC_CPU_IGP__00  EQPC_CPU_IGP__01
Addr: 0xb909000c        0xb909100c
Val:  0x00000111        0x000000f7
Name: EQPC_CPU_IGP__02  EQPC_CPU_IGP__03
Addr: 0xb909200c        0xb909300c
Val:  0x00000026        0x0000004a
Name: EQPC_DMA_IGP__04  EQPC_DMA_IGP__05
Addr: 0xb909400c        0xb909500c
Val:  0x00000000        0x000001eb
Name: EQPC_DMA_IGP__06  EQPC_DMA_IGP__07
Addr: 0xb909600c        0xb909700c
Val:  0x00000000        0x00000000
Name: EQPC_DMA_IGP__08  EQPC_DMA_IGP__09
Addr: 0xb909800c        0xb909900c

[...]
```

Figure 159 shows the interpretation of this output.

Figure 159 DMA Counter Printout

14.4.2.3 Queue Properties

The details of the configured queue configuration are shown in this example:

```
/sys/kernel/debug/pp/qos0# echo queue=41 > queue
[ 631.592923] [PP_QOS_DBG]:qos_dbg_props:304: Current configuration:
[ 631.599272] [PP_QOS_DBG]:qos_dbg_props:310: queue                          41
[ 631.608631] [PP_QOS_DBG]:qos_dbg_props:310: bw                             0
[ 631.617874] [PP_QOS_DBG]:qos_dbg_props:310: shared                         0
[ 631.627134] [PP_QOS_DBG]:qos_dbg_props:310: max_burst                      2
[ 631.636405] [PP_QOS_DBG]:qos_dbg_props:310: parent                         39
[ 631.645770] [PP_QOS_DBG]:qos_dbg_props:310: priority                       0
[ 631.655028] [PP_QOS_DBG]:qos_dbg_props:310: wrr_weight                     1
[ 631.664289] [PP_QOS_DBG]:qos_dbg_props:310: blocked                        0
[ 631.673547] [PP_QOS_DBG]:qos_dbg_props:310: wred_enable                    1
[ 631.682807] [PP_QOS_DBG]:qos_dbg_props:310: wred_fixed_drop_prob           0
[ 631.692067] [PP_QOS_DBG]:qos_dbg_props:310: min_avg_green                  115200
[ 631.701906] [PP_QOS_DBG]:qos_dbg_props:310: max_avg_green                  128000
[ 631.711743] [PP_QOS_DBG]:qos_dbg_props:310: slope_green                    100
[ 631.721235] [PP_QOS_DBG]:qos_dbg_props:310: fixed_drop_prob_green          0
[ 631.730494] [PP_QOS_DBG]:qos_dbg_props:310: min_avg_yellow                 57600
[ 631.740216] [PP_QOS_DBG]:qos_dbg_props:310: max_avg_yellow                 64000
[ 631.749939] [PP_QOS_DBG]:qos_dbg_props:310: slope_yellow                   100
[ 631.759430] [PP_QOS_DBG]:qos_dbg_props:310: fixed_drop_prob_yellow         0
[ 631.768690] [PP_QOS_DBG]:qos_dbg_props:310: min_guaranteed                 500
[ 631.778180] [PP_QOS_DBG]:qos_dbg_props:310: max_allowed                    8000
[ 631.787799] [PP_QOS_DBG]:qos_dbg_props:310: codel_en                       0
[ 631.797050] [PP_QOS_DBG]:qos_dbg_props:310: eir                            0
[ 631.806306] [PP_QOS_DBG]:qos_dbg_props:310: ebs                            0
```

14.4.2.4 Buffer Statistics

Print this to check the packet buffer segments and buffer pool statistics:

```
# cat /sys/kernel/debug/pp/buff_mgr/hw_stats
```

Figure 160 shows an example.

```
root@prx321-sfu-qspi-pon:/# cat /sys/kernel/debug/pp/buff_mgr/hw_stats

Control = 0x1
Pool Min Grant Bit Mask = 0x0
Pool Enable = 0xf000f
Pool FIFO Reset = 0xfff0
OCP Master Burst Size = 0xaa
OCP Master Number Of Bursts = 0x55
Status = 0x2

+=============================================================================================================+
|------------------------------------------  Pool Configuration  ------------------------------------------+
+======+=======+================+===========+================+==============+====================+==========+
| Pool | Size | FIFO Base Addr | FIFO Size | Prog Empty Thr | Prog Full Thr | Ext FIFO Base Addr | Isolated |
+------+------+----------------+-----------+----------------+---------------+--------------------+----------+
|0     |17536 |   0x       0   |256        |64              |192            |     0x23860000     |  False   |
|1     |1664  |   0x     100   |256        |64              |192            |     0x23854000     |  False   |
|2     |78080 |   0x     200   |256        |64              |192            |     0x23880000     |  False   |
|3     |6784  |   0x     300   |256        |64              |192            |     0x23858000     |  False   |
+======+======+================+===========+================+===============+====================+==========+

+=================================================================================================================+
|-----------------------------------------------  Pool Statistics  ------------------------------------------------+
+======+==========+=============+=========+=======+=======+==========+==========+=========+=========+
| Pool | FIFO Occ | Ext FIFO Occ |  Alloc  |  Pop  |  Push | Burst wr | Burst rd | Wm lthr | Wm lctr |
+------+----------+-------------+---------+-------+-------+----------+----------+---------+---------+
|0     |86        |16896        |253      |623    |370    |10        |0         |128      |0        |
|1     |82        |1280         |17       |17     |0      |6         |0         |128      |0        |
|2     |82        |77696        |17       |17     |0      |6         |0         |128      |0        |
|3     |82        |6400         |17       |17     |0      |6         |0         |128      |0        |
+=================================================================================================================+

+=============================================+
|---------------- Group Statistics ------------|
+=======+==================+==================+
| Group | Available Buffers | Reserved Buffers |
+-------+------------------+------------------+
+=======+==================+==================+

+=============================================================================================================+
|------------------------------------------  Policy Statistics  ------------------------------------------+
+========+=========+============+===============+==================+==============+====================+
| Policy | Null ctr | Max Allowed | Min Guaranteed | Group Association | Pools Mapping | Allocated Buffers |
+--------+----------+-------------+----------------+-------------------+---------------+--------------------+
|0       |0         |17536        |64              |Invalid            |0              |253                 |
|1       |0         |1664         |64              |Invalid            |1              |17                  |
|2       |0         |78080        |64              |Invalid            |2              |17                  |
|3       |0         |6784         |64              |Invalid            |3              |17                  |
+========+==========+=============+================+===================+===============+====================+

+================================================+
|------------  Policy/Pool Statistics  -----------+
+========+======+=============+====================+
| Policy | Pool | Max Allowed | Allocated Buffers |
+--------+------+-------------+--------------------+
|0       |0     |17536        |253                 |
|1       |1     |1664         |17                  |
|2       |2     |78080        |17                  |
|3       |3     |6784         |17                  |
+========+======+=============+====================+
```


Figure 160 Buffer Pool Statistics

14.4.3 GSWIP Core PCE Bypass Debugging

After QoS handling, the data packets are again forwarded through the GSWIP hardware, but not all processing stages are performed. This is called bypass mode.

The dedicated performance counters are available to probe the packet flow in this mode.

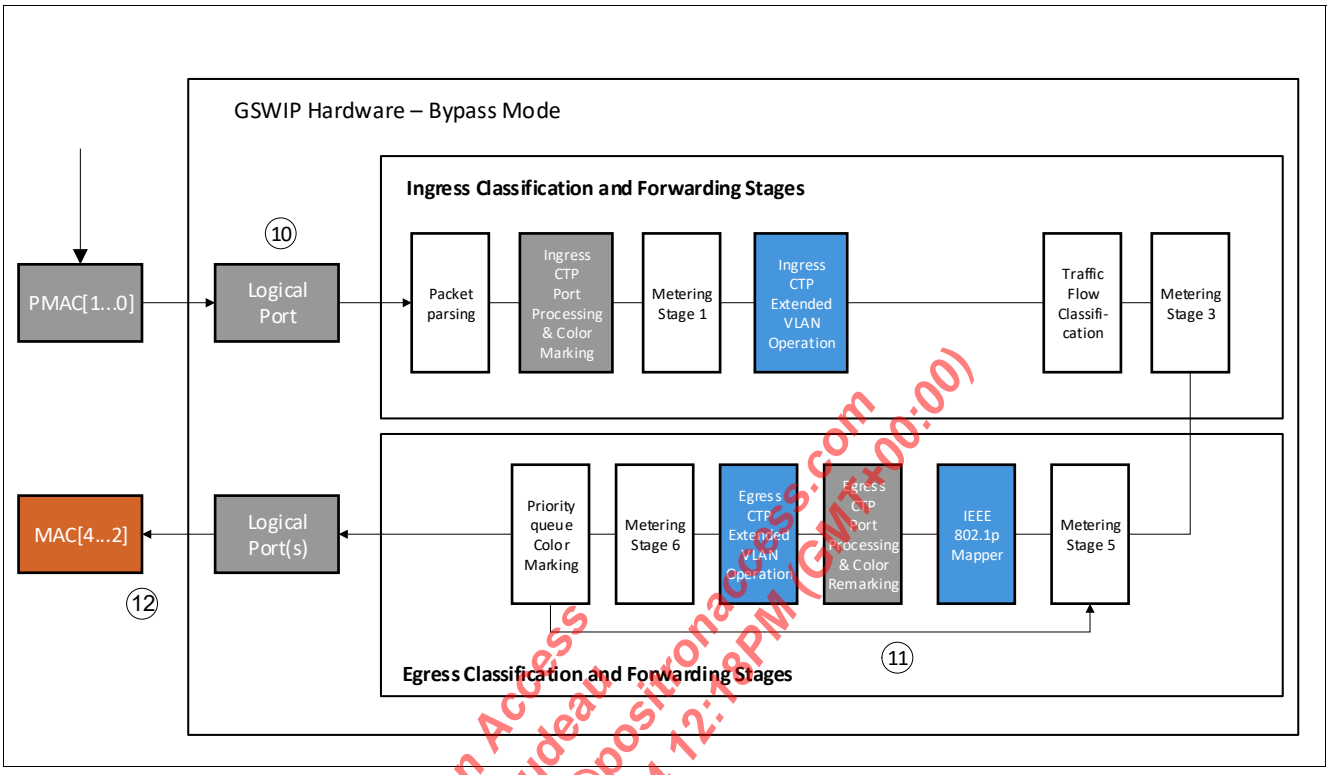


Figure 161 GSWIP Core Traffic Processing Stages in Bypass Mode

Probe Point 10: CTP Bypass Counters

Read the counters of a CTP with port ID=2 (the `nPortId` selector is identifying the ID of the CTP port).

```
# switch_cli GSW_CTP_PORT_PCEBYPASS_RMON_GET nPortId=2
GSW_CTP_PORT_PCEBYPASS_RMON_GET

nTxGoodPkts                    : 185
nTxUnicastPkts                 : 178
nTxBroadcastPkts               : 0
nTxMulticastPkts               : 7
nTx64BytePkts                  : 0
nTx127BytePkts                 : 3
nTx255BytePkts                 : 4
nTx511BytePkts                 : 0
nTx1023BytePkts                : 0
nTxMaxBytePkts                 : 0
nTxDroppedPkts                 : 0
nTxOversizeGoodPkts            : 0
nTxUnderSizeGoodPkts           : 178
nTxAcmDroppedPkts              : 0
nTxGoodBytes                   : 9754 (0x261a)
```

14.4.4 Extended VLAN Debugging

The extended VLAN configuration is configured on bridge or CTP ports. For each port, an extended VLAN block is assigned for ingress and egress. There are also extended VLAN blocks for IGMP and multicast for CTP ports. They are used only for multicast related traffic. Use these commands to check the configuration of a extVLAN block:

```
# switch_cli GSW_BRIDGE_PORT_CONFIG_GET nBridgePortId=3
        [...]

        nBridgePortId                                   = 0x3
        eMask                                           = 0xffffffff
        nBridgeId                                       = 0x2
        bIngressExtendedVlanEnable                      = 0x1
        nIngressExtendedVlanBlockId                     = 0x11
        bEgressExtendedVlanEnable                       = 0x1
        nEgressExtendedVlanBlockId                      = 0x1f

       [...]
```

The previous command shows that two extended VLAN blocks are attached to bridge port number 3 on ingress block `0x11` and egress block `0x1f`.

Use this command to read the entries contained in each block. The block ID must be specified together with the entry index within the block. When the index is not within the block then an error is returned. This command reads index `2` from block `0x11`.

```
# switch_cli GSW_EXTENDEDVLAN_GET nExtendedVlanBlockId=0x11 nEntryIndex=2
       [...]
        nExtendedVlanBlockId                    = 0
        nEntryIndex                             = 1
        eOuterVlanFilterVlanType                = 0
        bOuterVlanFilterPriorityEnable          = 0
        nOuterVlanFilterPriorityVal             = 8
        bOuterVlanFilterVidEnable               = 1
        nOuterVlanFilterVidVal                  = 3133
        eOuterVlanFilterTpid                    = 1
        eOuterVlanFilterDei                     = 0
        eInnerVlanFilterVlanType                = 3
        bInnerVlanFilterPriorityEnable          = 0
        nInnerVlanFilterPriorityVal             = 15
        bInnerVlanFilterVidEnable               = 0
        nInnerVlanFilterVidVal                  = 0
        eInnerVlanFilterTpid                    = 0
        eInnerVlanFilterDei                     = 0
        eEtherType                              = 0
        eRemoveTagAction                        = 1
        bOuterVlanActionEnable                  = 1
        eOuterVlanActionPriorityMode            = 1
        eOuterVlanActionPriorityVal             = 8
        eOuterVlanActionVidMode                 = 0
        eOuterVlanActionVidVal                  = 3133
        eOuterVlanActionTpid                    = 3
        eOuterVlanActioneDei                    = 2
        bInnerVlanActionEnable                  = 0
```

```
eInnerVlanActionPriorityMode          = 0
eInnerVlanActionPriorityVal           = 15
eInnerVlanActionVidMode               = 0
eInnerVlanActionVidVal                = 0
eInnerVlanActionTpid                  = 0
eInnerVlanActioneDei                  = 0
bNewDscpEnable                        = 0
nNewDscp                              = 0
bNewTrafficClassEnable                = 0
nNewTrafficClass                      = 0
bNewMeterEnable                       = 0
sNewTrafficMeterId                    = 0
bLoopbackEnable                       = 0
bDaSaSwapEnable                       = 0
bMirrorEnable                         = 0
bReassignBridgePortEnable             = 0
nNewBridgePortId                      = 0
```

An explanation of the previous fields is provided hereafter. Refer to the switch API documentation [2] for details.

```
eOuterVlanFilterVlanType: 0 normal, 1 no criteria, 2 default rule, 3 no tag, 4 inv.
bOuterVlanFilterPriorityEnable: filter on priority
nOuterVlanFilterPriorityVal: priority
bOuterVlanFilterVidEnable: Vid filter enable
nOuterVlanFilterVidVal: 0-4095: Vid 4096: any
eOuterVlanFilterTpid: 0: do not filter 1: 0x8100 2: global (0x88a8)
eOuterVlanFilterDei: 0: do not filter 1: DEI = 0 2: DEI = 1
eEtherType 0: no filter 1: IP 2: PPP 3: ARP 4: IPv6 5: EAPOL
eRemoveTagAction 0: no pop 1: 1: pop 1 tag 2: pop 2 tags 3: discard

eOuterVlanActionPriorityMode 0: set VID 1: copy from inner 2: copy from outer: 3:
copy from DSCP
eOuterVlanActionPriorityVal 0-7: pcp value 8: no pcp filter
eOuterVlanActionVidMode 0: VID 1: INNER_VID 2: OUTER_VID
eOuterVlanActionVidVal VID 0-4095
eOuterVlanActionTpid 0: copy inner 1: copy outer 2: global TPID 3: 802.1q
eOuterVlanActioneDei 0: copy inner 1: copy outer 2: DEI =0 3: DEI =1
```

An overall list of rules configured by using the `tc` command are dumped by calling:

```
cat /sys/kernel/debug/pon_qos/ext_vlan
```

14.4.5 VLAN Filter Debugging

The VLAN filter debugging is similar to debugging of the extended VLAN. There is a special block for each port identifying each VLAN filter block. These commands are shown as an example.

```
# switch_cli GSW_BRIDGE_PORT_CONFIG_GET nBridgePortId=9

        [...]
        nBridgePortId                                   = 0x9
        [...]
        nMacLearningCount                               = 0x0
        bIngressVlanFilterEnable                        = 0x1
        nIngressVlanFilterBlockId                       = 0x8
        bBypassEgressVlanFilter1                        = 0x0
        bEgressVlanFilter1Enable                        = 0x1
        nEgressVlanFilter1BlockId                       = 0x8
        bEgressVlanFilter2Enable                        = 0x0
        nEgressVlanFilter2BlockId                       = 0x0

# switch_cli GSW_VLANFILTER_GET nVlanFilterBlockId=0x8 nEntryIndex=0
        [...]
        nVlanFilterBlockId              = 8
        nEntryIndex                     = 0
        eVlanFilterMask                 = 0
        nVal                            = 3169
        bDiscardMatched                 = 0
```

The previous commands check for VLAN filter block with ID `8` and index `0`. The rule defines that the VLAN 3169 must pass.

14.4.6 UNI Interface Debugging

There are several options to check the UNI interface status.

Network Interface Status – ip

Check the network interfaces:

```
# ip -br l
lo               UNKNOWN        00:00:00:00:00:00 <LOOPBACK,UP,LOWER_UP>
ins0             UNKNOWN        00:00:00:00:00:00
<BROADCAST,MULTICAST,NOARP,UP,LOWER_UP>
dummy0           UNKNOWN        0a:d1:ff:94:83:51 <BROADCAST,NOARP,UP,LOWER_UP>
teql0            DOWN           <NOARP>
eth0_0           DOWN           00:e0:92:00:01:44 <BROADCAST,MULTICAST>
eth0_0_1_lct     UNKNOWN        00:e0:92:00:01:40 <BROADCAST,MULTICAST,UP,LOWER_UP>
eth0_0_2         DOWN           00:e0:92:00:01:44 <BROADCAST,MULTICAST>
eth0_0_3         DOWN           00:e0:92:00:01:44 <BROADCAST,MULTICAST>
eth0_0_us        DOWN           82:97:f2:35:88:7c <BROADCAST,MULTICAST>
eth0_1           DOWN           00:e0:92:00:01:45 <BROADCAST,MULTICAST>
eth0_1_1_lct     DOWN           ae:d2:6c:ae:b3:9f <BROADCAST,MULTICAST>
eth0_1_2         DOWN           00:e0:92:00:01:45 <BROADCAST,MULTICAST>
eth0_1_3         DOWN           00:e0:92:00:01:45 <BROADCAST,MULTICAST>
eth0_1_us        DOWN           ea:b9:ac:5d:94:83 <BROADCAST,MULTICAST>
pon0             DOWN           00:00:00:00:00:00 <BROADCAST,MULTICAST>
```

```
tcont-omci@pon0   DOWN            3e:60:8e:2e:16:f4 <BROADCAST,MULTICAST,M-DOWN>
gem-omci@pon0    UNKNOWN       00:e0:92:00:01:42 <BROADCAST,MULTICAST,UP,LOWER_UP,M-
DOWN>
iphost1@pon0      DOWN            8a:7a:96:65:67:3b <BROADCAST,MULTICAST,M-DOWN>
```

Network Interface Status – ifconfig

Check the network interfaces:

```
# ifconfig
dummy0    Link encap:Ethernet HWaddr DA:9D:9C:2D:22:C2
          UP BROADCAST RUNNING NOARP MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B) TX bytes:276 (276.0 B)

eth0_0_1_lct Link encap:Ethernet HWaddr 00:E0:92:00:01:40
          inet addr:10.17.17.190 Bcast:10.17.17.255 Mask:255.255.255.0
          inet6 addr: fe80::2e0:92ff:fe00:140/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:62367 errors:0 dropped:14 overruns:0 frame:0
          TX packets:85 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7050131 (6.7 MiB) TX bytes:5498 (5.3 KiB)

gem-omci  Link encap:Ethernet HWaddr 00:E0:92:00:01:42
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B) TX bytes:138 (138.0 B)

host      Link encap:Ethernet HWaddr 00:E0:92:00:01:41
          inet6 addr: fe80::2e0:92ff:fe00:141/64 Scope:Link
          UP BROADCAST RUNNING NOARP MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```


Ethernet Interface Status on LAN side – ethtool

Check the status of each of the physical Ethernet network interfaces.

```
# ethtool eth0_0
Settings for eth0_0:
        Supported ports: [ TP MII ]
        Supported link modes:   100baseT/Full
                                1000baseT/Full
                                10000baseT/Full
        Supported pause frame use: No
        Supports auto-negotiation: Yes
        Supported FEC modes: Not reported
        Advertised link modes: 100baseT/Full
                                1000baseT/Full
                                10000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Advertised FEC modes: Not reported
        Speed: 1000Mb/s
        Duplex: Full
        Port: MII
        PHYAD: 0
        Transceiver: internal
        Auto-negotiation: on
        Link detected: no

# ethtool eth0_1
Settings for eth0_1:
        Supported ports: [ TP MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
                                2500baseT/Full
        Supported pause frame use: Symmetric
        Supports auto-negotiation: Yes
        Supported FEC modes: Not reported
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
                                2500baseT/Full
        Advertised pause frame use: Symmetric
        Advertised auto-negotiation: Yes
        Advertised FEC modes: Not reported
        Link partner advertised link modes:  10baseT/Half 10baseT/Full
                                             100baseT/Half 100baseT/Full
        Link partner advertised pause frame use: No
        Link partner advertised auto-negotiation: No
        Link partner advertised FEC modes: Not reported
        Speed: 100Mb/s
        Duplex: Full
        Port: MII
        PHYAD: 31
        Transceiver: internal
        Auto-negotiation: on
```

```
        Link detected: no
```

These `ethtool` options are supported:

- `-s`
- `-a`
- `-A`
- `-k`
- `-i`
- `-r`
- `-S`
- `-T`
- `-P`
- `--show-priv-flags`
- `--set-priv-flags`
- `--show-eee`
- `--set-eee`
- `-h`
- `--version`

These private flags are supported:

- `bp_to_cpu_enable`: Connect the CPU port to a bridge.

Note: Other options or private flags are not supported, even when listed in the help printout.

XPCS Status

For example:

```
# cat /sys/devices/platform/1a000000.ssx1_2/1a342000.lan_xpcs0/xpcs_linksts

lan_xpcs0: LINK Status


PMA Tx status

        TX_ACK_0:                       Tx NOT Acknowledge on Lane 0
        Rx Detection Result on lane 0:  Rx NOT Detected on Lane 0
        TX0_PSTATE:                     00000000

PMA Rx status

        PMA_STS:                        LINK UP
        PMA_STATUS2 RF:                 Rx No Fault
        PMA_KR_PMD_STS RCV_STS:         Rx training
        PMA_KR_PMD_STS FRM_LCK:        Training frame delineation not detected
        PMA_KR_PMD_STS SU_PR_DTD:       Start-up protocol complete
        PMA_KR_PMD_STS TR_FAIL:         Training is not yet complete

10GBASE-KR LP coeff Update

        PMA_KR_PMD_STS LP_INIT:         Normal operation
        PMA_KR_PMD_STS LP_PRST:         Normal operation

10GBASE-KR LP coeff Status
```

```
        PMA_KR_LP_CESTS LP_RR:                The LP rx is requesting that the
                                              training should be continued.

10GBASE-KR LD coeff Update Local Device requests to Link Partner.

        PMA_KR_LD_CEU LD_INIT:                Normal operation
        PMA_KR_LD_CEU PRST:                   Normal operation

10GBASE-KR LD coeff Status

       PMA_KR_LD_CESTS RR:                The LD rx is requesting that the training
should be continued.

Autonegotiation Status Register

        LP AN Ability:                        LP unable to participate in AN
        AN Link Status:                       CL73 not complete and no valid link
        LD AN Ability:                        LD supports CL73 AN
        AN Remote Fault:                      No fault detected
        AN Complete:                          AN not complete
        Page Received:                        No page received
        Parallel Detection Fault in CL73 :    No fault detected

PCS Rx Status

        PCS_STS:                              LINK UP
        PCS_STATUS2 RF:                       Rx No Fault
        PCS_STATUS2 TF:                       Tx No Fault
        CAP_EN:                               10GBASE-R Capable
        CAP_10_1GC:                           10GBASE-X Capable
        CAP_10GBW:                            10GBASE-W Not Capable
        CAP_10GBT:                            10GBASE-T Not Capable
       DS:                                 MMD is present and responding to this reg addr
        PCS_KR_STS2 :                         80008000
        PCS_TYPE_SEL:                         10GBASE-R PCS Type

MII Status

        1000BASE-X mode Rx:                   LINK DOWN
        SGMII/QSGMII/USXGMII mode Port0 Rx:   LINK DOWN
```


14.4.7 ANI Interface Debugging

There are several options to check the ANI interface status.

Network Interface Status – ip

Check the network interfaces:

```
# ip -br l | grep pon
pon0              DOWN           00:00:00:00:00:00 <BROADCAST,MULTICAST>
tcont-omci@pon0   DOWN           2e:7f:f9:88:cd:6d <BROADCAST,MULTICAST,M-DOWN>
```

```
gem-omci@pon0     UNKNOWN         02:50:f1:00:00:01
<BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP,M-DOWN>
```

Network Interface Details

Use this command to inspect the details of network interfaces:

`ip -d l show type <interface type>`

with `<interface type>` to be set to `gem`, `pon`, `tcont`, `pmapper`, or `iphost`.

Ethernet Interface Status on ANI side – ethtool

The PMD status information can be read from the PMD by using the ethtool command on the `pon0` network interface. For example:

```
# ethtool -m pon0
  Identifier                                : 0x03 (SFP)
  Extended identifier                       : 0x04 (GBIC/SFP defined by 2-wire
interface ID)
  Connector                                 : 0x01 (SC)
  Transceiver codes                    : 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
  Encoding                                  : 0x03 (NRZ)
  BR, Nominal                               : 9900MBd
  Rate identifier                           : 0x00 (unspecified)
  Length (SMF, km)                          : 20km
  Length (SMF)                              : 20000m
  Length (50um)                             : 0m
  Length (62.5um)                           : 0m
  Length (Copper)                           : 0m
  Length (OM3)                              : 0m
  Laser wavelength                          : 1270nm
  Vendor name                               : SUPERXON LTD.
  Vendor OUI                                : 00:00:00
  Vendor PN                                 : SOGX2699-PSGA
  Vendor rev                                : 10
  Option values                             : 0x00 0x1c
  Option                                    : RX_LOS implemented, inverted
  Option                                    : TX_FAULT implemented
  Option                                    : TX_DISABLE implemented
  BR margin, max                            : 0%
  BR margin, min                            : 0%
  Vendor SN                                 : 97961183200205
  Date code                                 : 180817
  Optical diagnostics support               : Yes
  Laser bias current                        : 24.430 mA
  Laser output power                        : 1.7784 mW / 2.50 dBm
  Receiver signal average optical power     : 0.0001 mW / -40.00 dBm
  Module temperature                        : 51.54 degrees C / 124.78 degrees F
  Module voltage                            : 3.2999 V
  Alarm/warning flags implemented           : Yes
  Laser bias current high alarm             : Off
  Laser bias current low alarm              : Off
  Laser bias current high warning           : Off
  Laser bias current low warning            : Off
```

```
Laser output power high alarm             : Off
Laser output power low alarm              : Off
Laser output power high warning           : Off
Laser output power low warning            : Off
Module temperature high alarm             : Off
Module temperature low alarm              : Off
Module temperature high warning           : Off
Module temperature low warning            : Off
Module voltage high alarm                 : Off
Module voltage low alarm                  : Off
Module voltage high warning               : Off
Module voltage low warning                : Off
Laser rx power high alarm                 : Off
Laser rx power low alarm                  : On
Laser rx power high warning               : Off
Laser rx power low warning                : On
Laser bias current high alarm threshold   : 50.000 mA
Laser bias current low alarm threshold    : 0.000 mA
Laser bias current high warning threshold : 45.000 mA
Laser bias current low warning threshold  : 0.000 mA
Laser output power high alarm threshold   : 5.0000 mW / 6.99 dBm
Laser output power low alarm threshold    : 0.9976 mW / -0.01 dBm
Laser output power high warning threshold : 3.9716 mW / 5.99 dBm
Laser output power low warning threshold  : 1.2559 mW / 0.99 dBm
Module temperature high alarm threshold   : 75.00 degrees C / 167.00 degrees F
Module temperature low alarm threshold    : -5.00 degrees C / 23.00 degrees F
Module temperature high warning threshold : 70.00 degrees C / 158.00 degrees F
Module temperature low warning threshold  : 0.00 degrees C / 32.00 degrees F
Module voltage high alarm threshold       : 3.6000 V
Module voltage low alarm threshold        : 3.0000 V
Module voltage high warning threshold     : 3.5000 V
Module voltage low warning threshold      : 3.1000 V
Laser rx power high alarm threshold       : 0.1995 mW / -7.00 dBm
Laser rx power low alarm threshold        : 0.0010 mW / -30.00 dBm
Laser rx power high warning threshold     : 0.1585 mW / -8.00 dBm
Laser rx power low warning threshold      : 0.0013 mW / -28.86 dBm
```

14.4.8 Multicast Debugging

The multicast daemon provides a command line interface used to read the multicast forwarding configuration and to modify it. Use this to show the available commands:

```
# ponmcc --help
PON MCC driver command line interface

Parameters:
--table_entry_add --ifindex {ifindex} --proto {proto (0 - IPv4, 1 - IPv6)} --
grp_addr {grp_addr} --src_addr {src_addr} --flt_mode {flt_mode}
--controlled {controlled}

--table_entry_delete --ifindex {ifindex} --proto {proto (0 - IPv4, 1 - IPv6)} --
grp_addr {grp_addr} --src_addr {src_addr} --flt_mode {flt_mode} --
controlled {controlled}

--table_entry_read {initial}

--multicast_enable --igmp_version {igmp_version} --mc_gem_ifindex {mc_gem_ifindex}

--multicast_disable
```

The multicast group addresses are given, for example, as `224.1.0.1`.

To list the contents of the multicast forwarding table the `switch_cli` commands can be used:

```
switch_cli GSW_MULTICAST_TABLE_ENTRY_READ
```

14.4.9 MTU Debugging

The system MTU size (maximum packet size to be received and transmitted) is controlled through OMCI. The initial value is defined in the MIB initialization file, for example a value of 1534 in `/etc/mibs/prx300_1U.ini`.

This is checked by:

```
# grep "11 0x" /etc/mibs/prx300_1U.ini
11 0x0101 0 0 0 0x00 1 1 0 1534 0 0xffff 0 0 0 0 0
```

This value is changed by the OLT after the MIB initialization.

To check the value set by the OLT and assuming the ME ID is 257 for the PPTP Ethernet UNI, the (optionally modified) OMCI MIB value is read by:

```
# omci meg 11 257 | grep Max -A 1
 8 Max frame size                2b UINT RW---------
   0x05fe (1534)
```

To modify the MTU size outside the OMCI during manual debugging, use the `ip` command to set the value directly:

```
ip link set dev eth0_0 mtu 3600
```

The value is also read back directly from the hardware by calling:

```
# switch_cli gswss get mtu 3
gswss
GSWSS: MAC3 MTU: 3622
```

14.5 1PPS Debug Function

There are two CLI commands provided to enable the 1PPS output signal in case the OLT does not enable it by using the appropriate OLT-G managed entity configuration. The CLI commands to be used are:

- `pon tod_cfg_set`
  Configures the 1PPS pulse properties and enables the debug mode (free-running mode).
- `pon gpon_tod_sync_set`
  Defines the synchronization condition (multiframe/SFC counter value vs time value)

Complete these steps to test the 1PPS output signal without OLT synchronization.

1. Connect the ONU to an OLT.
2. Make sure the connection is established, at least PLOAM state O5 must be reached.
3. Initially configure the output signal properties by calling `pon tod_cfg_set` with:
   a) `pps_width` (default: use 500, high level time in ms)
   b) `pps_level` (default: use 1, raising edge on the1PPS event)
   c) `pps_scale` (default: use 1, do not divide the frequency)
   d) `pps_delay` (default: use 0, delay time in ps)
   e) `pps_debug` (default: use 0, keep the debug mode disabled during the initialization)
   f) `capture_edge` (default: use 0, this is not related to 1PPS)
   g) `us_ts_corr` (default: use 0, this is not related to 1PPS)
   h) `ds_ts_corr` (default: use 0, this is not related to 1PPS)
4. Write the virtual synchronization condition by calling `pon gpon_tod_sync_set` with:
   a) `multiframe_count` = 0xFFFF FFFF. Possibly a smaller value, but must be higher than the multiframe number received from the OLT at the time the debug function is enabled.
   b) `tod_seconds` = any time value, even 0
   c) `tod_extended_seconds` = 0
   d) `tod_nano_seconds` = any time value, must be larger than 3
   e) `tod_offset_pico_seconds` = any time value, even 0
   f) `tod_quality` = 0. This is read-only.
5. Enable the free running debug mode by again calling `pon tod_cfg_set` with:
   a) `pps_debug` = 1 (other values unchanged)
   Enables the free-running mode and triggers one single synchronization event.

The debug enable command is called again with different synchronization settings, each time a trigger event is generated to update the configured time value. For example:

```
pon gpon_tod_sync_set 0xFFFFFFFF 0 0 0 0 0
pon tod_cfg_set 10 1 1 0 1 0 0 0
```

Attention: A new trigger event is only generated when the trigger has been armed by writing a compare value using pon tod_cfg_set.

To disable the free-running mode, use `pon tod_cfg_set` with `pps_debug` = 0.

After that, the internal hardware waits for the next multiframe match, as configured through `pon gpon_tod_sync_set`.

See **Switch from UART0 to UART1**.

14.6 Debug Security

Debug access to the hardware and firmware is an inherent security risk. To overcome this, secure debug operation has been implemented and is recommended for use.

To use this functionality, the Secure Boot mode must be enabled. During productive operation, debug functionality is disabled and end users are not able to operate debug commands.

For system debugging, the debug functions are enabled by supplying the credentials to the trusted environment platform (TEP). For details, refer to the UGW documentation **[16]**, **[22]**.

14.6.1 Secure PON Debugging

Several debug functions are only available when secure debugging is enabled and not during normal operation.

When used from the command line, an error message is returned, indicating the desired function is not available.

When used from the software source code (which must be avoided), an error code is returned and must be covered by the usual return code handling.

14.7 Rogue ONU Testing

There are two different rogue detection methods supported by the device:

- Rogue ONU detection by the OLT
- Local rogue ONU detection

Both methods are tested individually.

14.7.1 Rogue ONU Detection by the OLT

To verify the combined OLT and ONU system behavior in case of a rogue ONU situation, the debug function `fapi_pon_debug_rogue_start` puts the ONU into a rogue state. The command line short form is `pon drs`.

This forces a continuous electrical upstream signal to the PMD and enables the optical transmission. During assigned T-CONT time slots, regular data is transmitted. Outside of these time slots, a fixed pattern is sent, as defined in `/etc/config/optic` as `burst_idle_pattern`.

Note: Ensure that the burst idle pattern is not all-zero.

The OLT is expected to detect this and start its rogue ONU mitigation procedure. By then, the ONU must be either disabled or re-enabled using PLOAM messages, depending on the OLT handling.

The function `pon fapi_pon_debug_rogue_stop` terminates this test mode.

Upon reconnection with the OLT, the test mode is automatically terminated.

This is a debug function and it is not available when secure debugging is *disabled*. This avoids unintentional rogue ONU states in a productive system.

Attention: Do not activate this test in a live system as it breaks all other ONU connections on the same OLT output (ODN).

14.8 Optical Transmitter Calibration

For production calibration purposes of the transmit timing, the transmitter is fed by a continuous PRBS signal.

The PRBS signal is generated by the SerDes between PON IP and the optical interface to the PMD.

In addition, the optical transceiver or PMD device used by the application must be set to continuous transmission mode. The exact operation depends on the optical device type and must be controlled by implementer-specific software through I^2C register access.

This is a debug function which is not available when secure debugging is disabled. This avoids unintentional rogue ONU states in the system.

A script is available at `/usr/bin/pon_optic_cal.sh` to ease the handling of the required settings. See **[10]** for more detail on optical interface timing calibration.

Note: For calibration of the laser driver and optical receiver, please contact the vendor of the optical module or PMD/BOSA devices.

14.9 SerDes Configuration

There are three available SerDes instances used to receive and transmit data on.

- WAN interface, either
  - PON or
  - Ethernet
- LAN interface #0
- LAN interface #1

The test functionality includes:

- Test pattern transmission (PRBS or fixed patterns)
- Test pattern reception (PRBS and fixed pattern)
- Bit error insertion
- Bit error counting

These test pattern types are supported:

- PPRBS pattern X^23 + X^18 + 1
- RBS pattern X^31 + X^28 + 1
- PRBS pattern X^23 + X^21 + X^16 + X^8 + X^5 + X^2 + 1
- PRBS pattern X^16 + X^5 + X^4 + X^3 + 1
- PRBS pattern X^15 + X^14 + 1
- PRBS pattern X^11 + X^9 + 1
- PRBS pattern X^9 + X^5 + 1
- PRBS pattern X^7 + X^6 + 1
- Repeated fixed 8-bit pattern (normal, inverse, or DC-balanced alternating)

14.9.1 Ethernet SerDes Configuration

The Ethernet SerDes test functions are available through command line access as shown in this example:

```
# cd /sys/devices/platform/1a000000.ssx1_2/1a342000.lan_xpcs0

# echo help tx_test_pattern_enable > bert
Usage:
  echo <cmd> > bert
  cmd:
    tx_test_pattern_enable:  Enable Tx pattern
    tx_test_pattern_disable:  Disable Tx test pattern
    rx_test_pattern_enable:  Enable Rx test pattern
    rx_test_pattern_disable:  Disable Rx test pattern
    remote_loop_enable:  Enable parallel Rx to Tx loop
    remote_loop_disable:  Disable parallel Rx to Tx loop
    local_loop_enable:  Enable serial Rx to Tx loop
    local_loop_disable:  Disable serial Rx to Tx loop
    tx_test_pattern_cfg_set:  Define the pattern to be sent (PRBS/fixed pattern)
    tx_test_pattern_cfg_get:  Get Tx BERT mode
    rx_test_pattern_cfg_set:  Define the pattern to be received (PRBS/fixed pattern)
    rx_test_pattern_cfg_get:  Get Rx BERT mode
    rx_bit_error_counter_get: Read the bit error counter, clear-on-read
```

Receive Test Pattern usage

These receive test patterns are used by applying the previously mentioned commands.

PRBS-9 test pattern reception:

```
# echo rx_test_pattern_cfg_set prbs9 >
/sys/devices/platform/12000000.ssx1_0/subsystem/devices/1a342000.lan_xpcs0/bert

# echo rx_test_pattern_cfg_get >
/sys/devices/platform/12000000.ssx1_0/subsystem/devices/1a342000.lan_xpcs0/bert
[…]          TX bert mode = BERT_PRBS9_TX
```

PRBS-31 test pattern reception:

```
# echo rx_test_pattern_cfg_set prbs31 >
/sys/devices/platform/12000000.ssx1_0/subsystem/devices/1a342000.lan_xpcs0/bert

# echo rx_test_pattern_cfg_get >
/sys/devices/platform/12000000.ssx1_0/subsystem/devices/1a342000.lan_xpcs0/bert
[…]          TX bert mode = BERT_PRBS31_RX
```

Transmit Test Pattern usage

These transmit test patterns are used by applying the previously mentioned commands.

PRBS-9 test pattern transmission:

```
# echo tx_test_pattern_cfg_set prbs9 >
/sys/devices/platform/12000000.ssx1_0/subsystem/devices/1a342000.lan_xpcs0/bert

# echo tx_test_pattern_cfg_get >
/sys/devices/platform/12000000.ssx1_0/subsystem/devices/1a342000.lan_xpcs0/bert
[…]          TX bert mode = BERT_PRBS9_TX
```

PRBS-31 test pattern transmission:

```
# echo tx_test_pattern_cfg_set prbs31 >
/sys/devices/platform/12000000.ssx1_0/subsystem/devices/1a342000.lan_xpcs0/bert

# echo tx_test_pattern_cfg_get >
/sys/devices/platform/12000000.ssx1_0/subsystem/devices/1a342000.lan_xpcs0/bert
[…]          TX bert mode = BERT_PRBS31_TX
```

Square wave test pattern transmission:

```
# echo tx_test_pattern_cfg_set square >
/sys/devices/platform/12000000.ssx1_0/subsystem/devices/1a342000.lan_xpcs0/bert

# echo tx_test_pattern_cfg_get >
/sys/devices/platform/12000000.ssx1_0/subsystem/devices/1a342000.lan_xpcs0/bert
[…]          TX bert mode = BERT_SQUAREWAVE_TX
```

14.9.2 PON SerDes Configuration

There are dedicated PON library functions available to generate and receive test patterns between the PON-side SerDes hardware and a remote test pattern generator/receiver. These functions are available only when the PON debug mode is enabled, because some of the functions disturb other data traffic on the PON (rogue ONU behavior). See **Section 14.3.3** for details.

14.9.2.1 TX Equalizer Settings

To achieve the best optical TX signal, the waveform of the electrical TX signal is adapted to board or PMD specific characteristics. The TX equalizer settings are stored in the configuration file `/etc/config/serdes` and read once after boot. Pre-amplitude, post-amplitude, DC swing, and max swing are controlled by parameters within the `/etc/config/serdes` file.

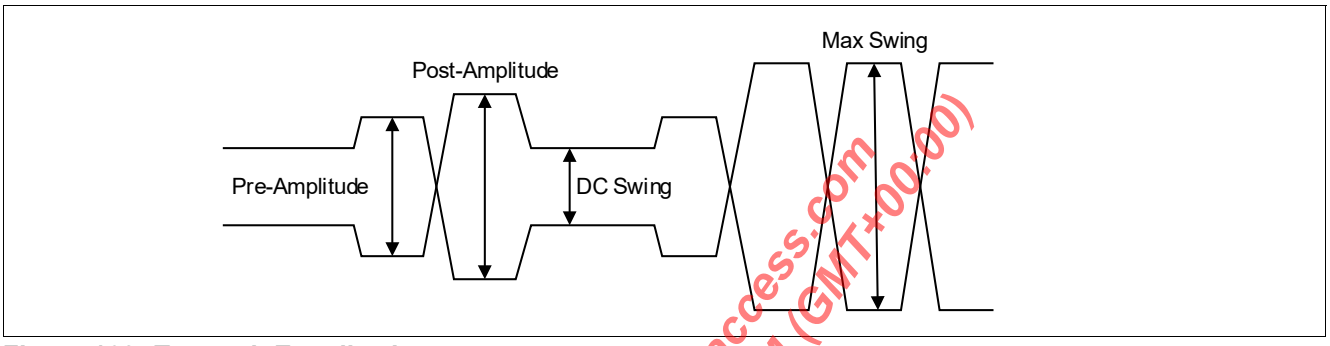


Figure 162 Transmit Equalization

The amplitudes pre-amplitude, post-amplitude and DC swing are calculated from the FFE coefficients as follows:

- Pre-amplitude = Vswing * (tx_eq_main + tx_eq_pre/4 - tx_eq_post/4) / 40
- Post-amplitude = Vswing * (tx_eq_main - tx_eq_pre/4 + tx_eq_post/4) / 40
- DC swing = Vswing * (tx_eq_main - tx_eq_pre/4 - tx_eq_post/4) / 40
- Max swing = Vswing * (tx_eq_main + tx_eq_pre/4 + tx_eq_post/4) / 40

with Vswing equal to the supply voltage of 0.9 V.

For maximum voltage swing, set (tx_eq_main + tx_eq_pre/4 + tx_eq_post/4) = 40.

For 50% voltage swing, set (tx_eq_main + tx_eq_pre/4 + tx_eq_post/4) = 20.

For 10G upstream data rates, the optimum optical TX eye has been achieved with the values in **Table 93**. These values may vary depending on the PCB version or PMD/BOSA type.

Table 93 Transmit Equalizer Settings

Parameter in /etc/config/serdes	Value for 10G Upstream Rate, EASY PRX321 V1.4 (SFU)
vboost_en	0
vboost_lvl	5
iboost_lvl	15
tx_eq_pre	0
tx_eq_main	40
tx_eq_post	0

14.9.2.2 Optical Interface Test and Loopback Functions

The CPE Software Suite provides multiple loopback, PRBS testing, and inverter functions based on the SerDes hardware test features as illustrated in **Figure 163**.

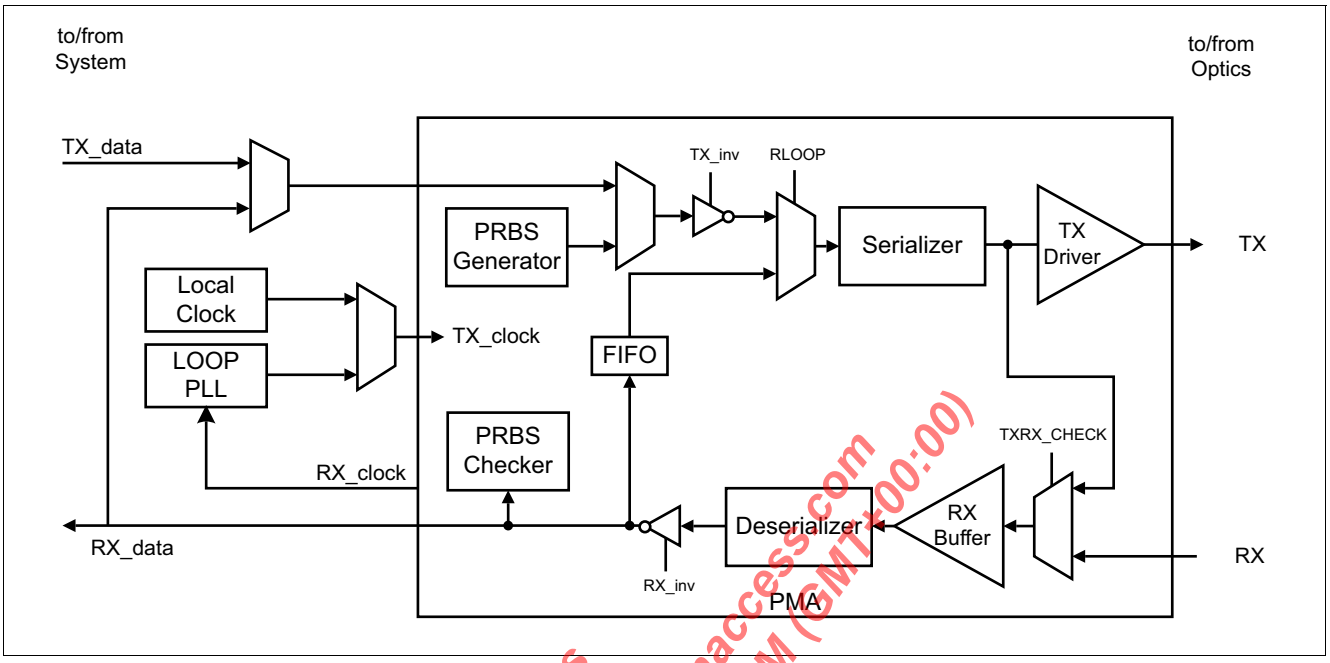


Figure 163 Optical Interface Test Pattern, Loopback, and Inverter Functions

14.9.2.2.1 Test Pattern Configuration

The function `debug_test_pattern_cfg_set/get` allows the test pattern used for PRBS generation and checking to be configured. It allows PRBS31, PRBS23, PRBS16, PRBS15, PRBS11, PRBS9, PRBS7, and fixed test pattern to be configured independently for the receive and the transmit direction.

14.9.2.2.2 Test Pattern Mode Control

After the test pattern has been configured, the function `debug_test_pattern_enable <test_mode> <wl_ch_id>` starts the test pattern generation and/or checking with the options indicated in `<mode>`:

- 1: RX
  PRBS checking is enabled.
- 2: TX_LOOP
  PRBS generation is enabled, with TX rate derived from loop timing.
- 3: TX_LOCAL
  PRBS generation is enabled, with TX rate derived from local XTAL timing.
- 4: TXRX_LOOP
  PRBS generation and checking are enabled, with loop timing.
- 5: TXRX_CHECK
  PRBS generation and checking are enabled. Generated PRBS is transmitted, serialized, and looped back to serial receiver, de-serialized, and PRBS-checked.
- 6: PMA_RLOOP
  Received data is looped back to the transmit path within PMA/SerDes (loop PLL bypass).
- 7: PON_RLOOP
  Received data is looped back to the transmit path within PON TC layer (loop PLL enabled).
  In parallel, PRBS checking is enabled.

Note: The transmit wavelength channel must be selected.
Use 0 if only a single upstream wavelength is provided by the optical transmitter.

When the pattern checker is enabled, it must be polled and cleared periodically.

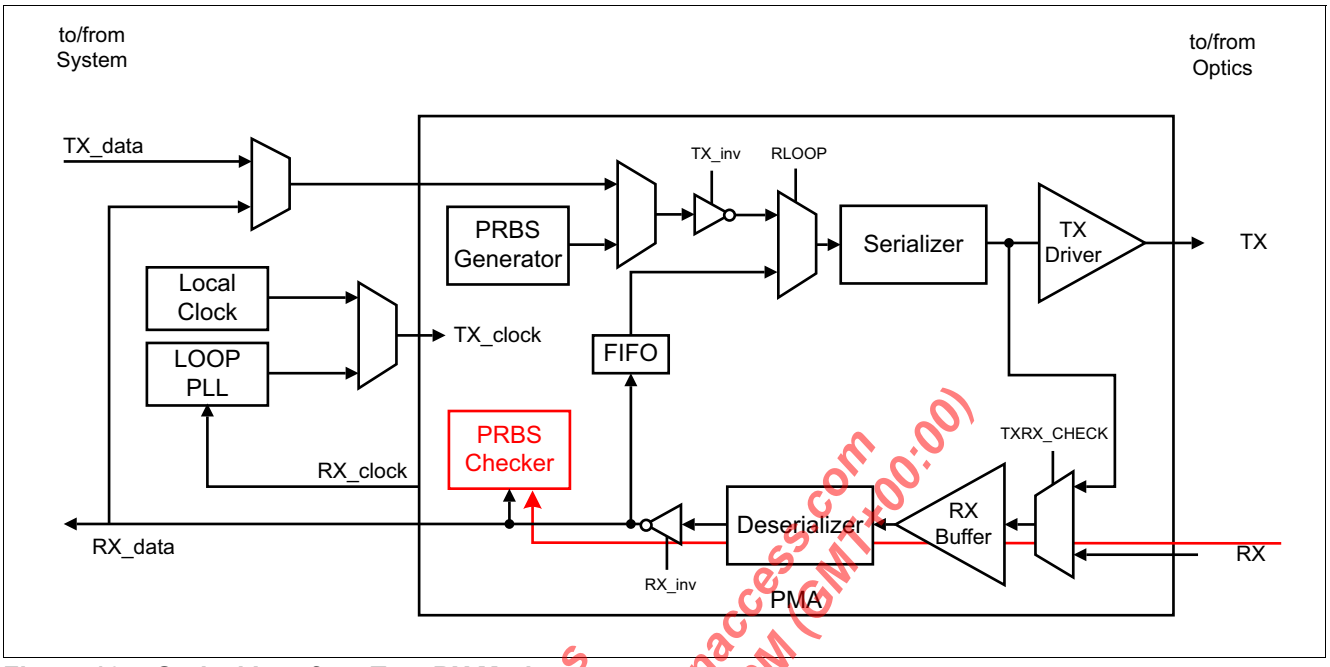


Figure 164 Optical Interface Test RX Mode

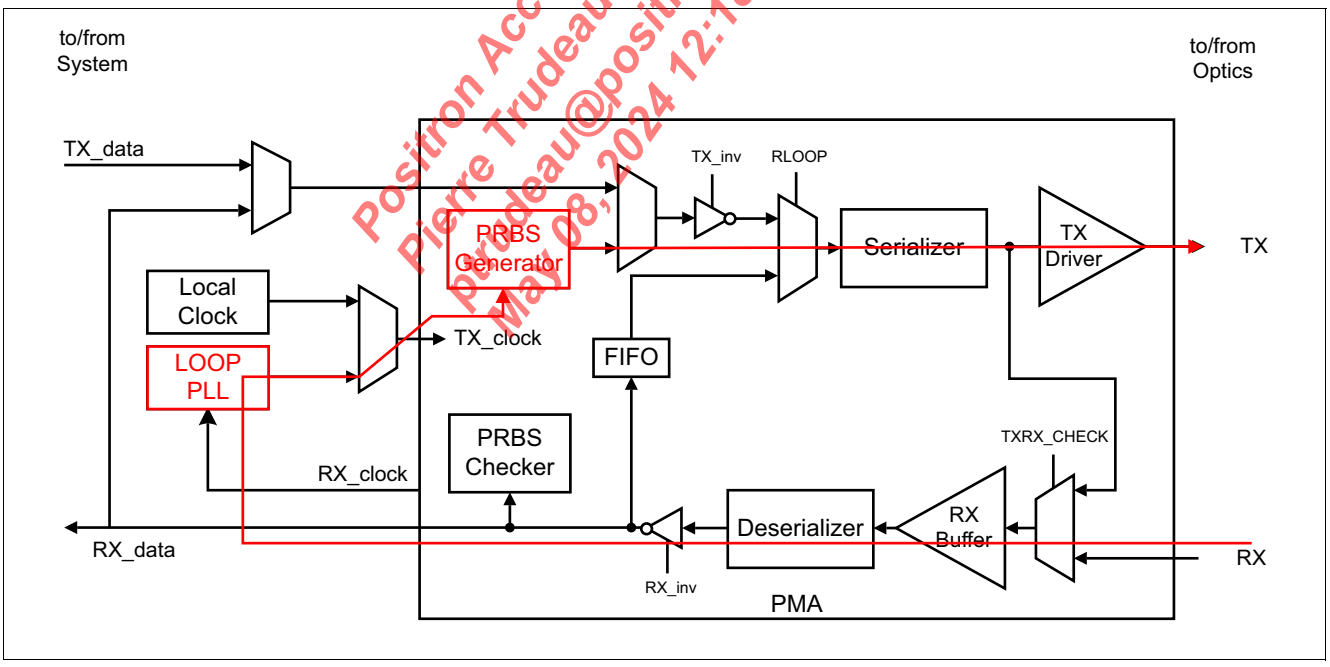


Figure 165 Optical Interface Test TX_LOOP Mode

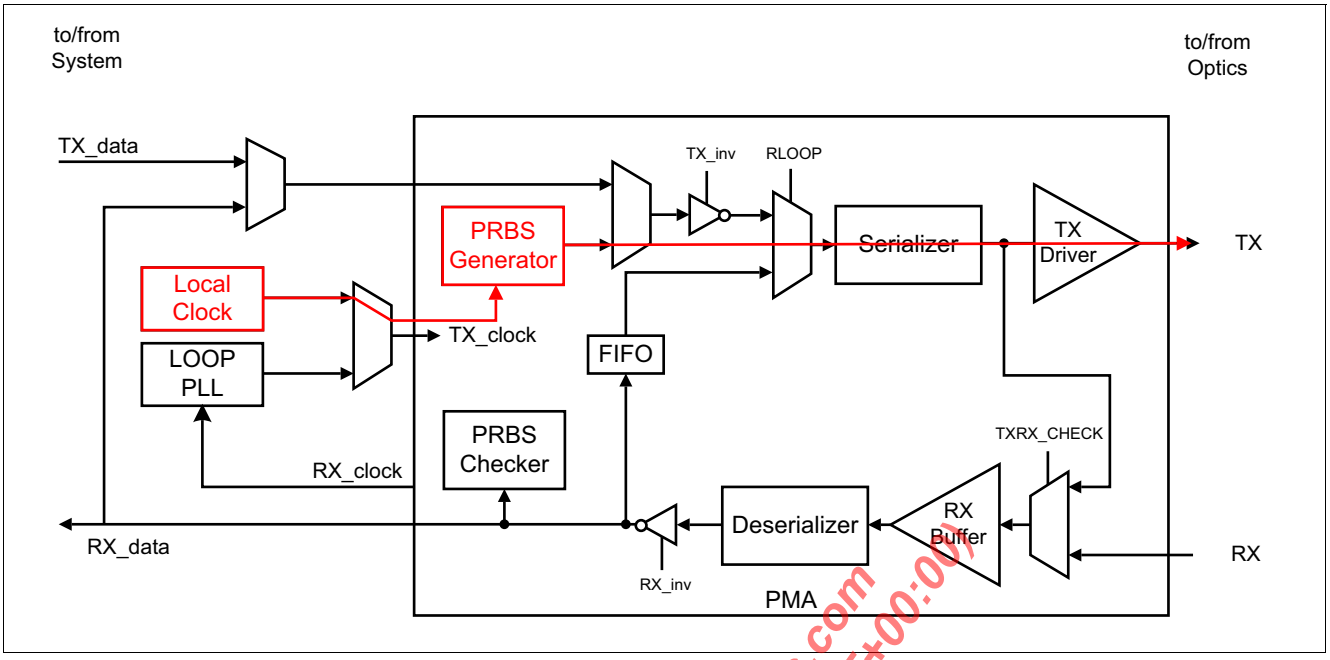


Figure 166 Optical Interface Test TX_LOCAL Mode

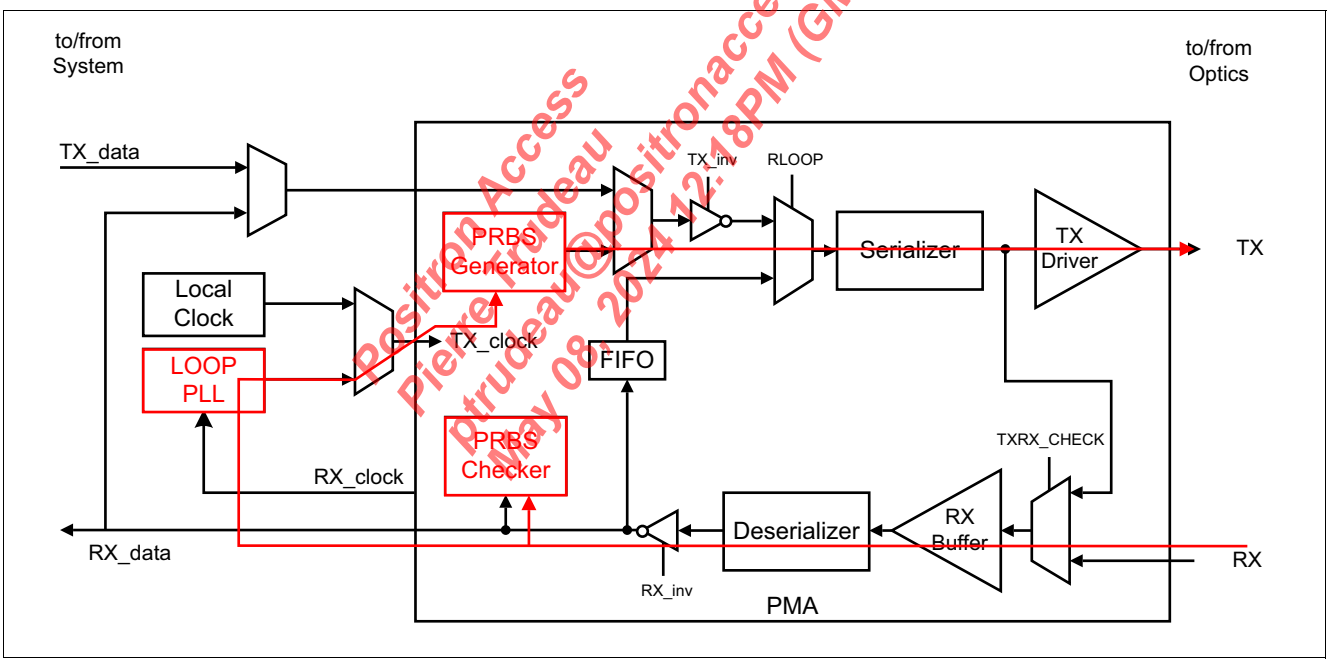


Figure 167 Optical Interface Test TXRX_LOOP Mode

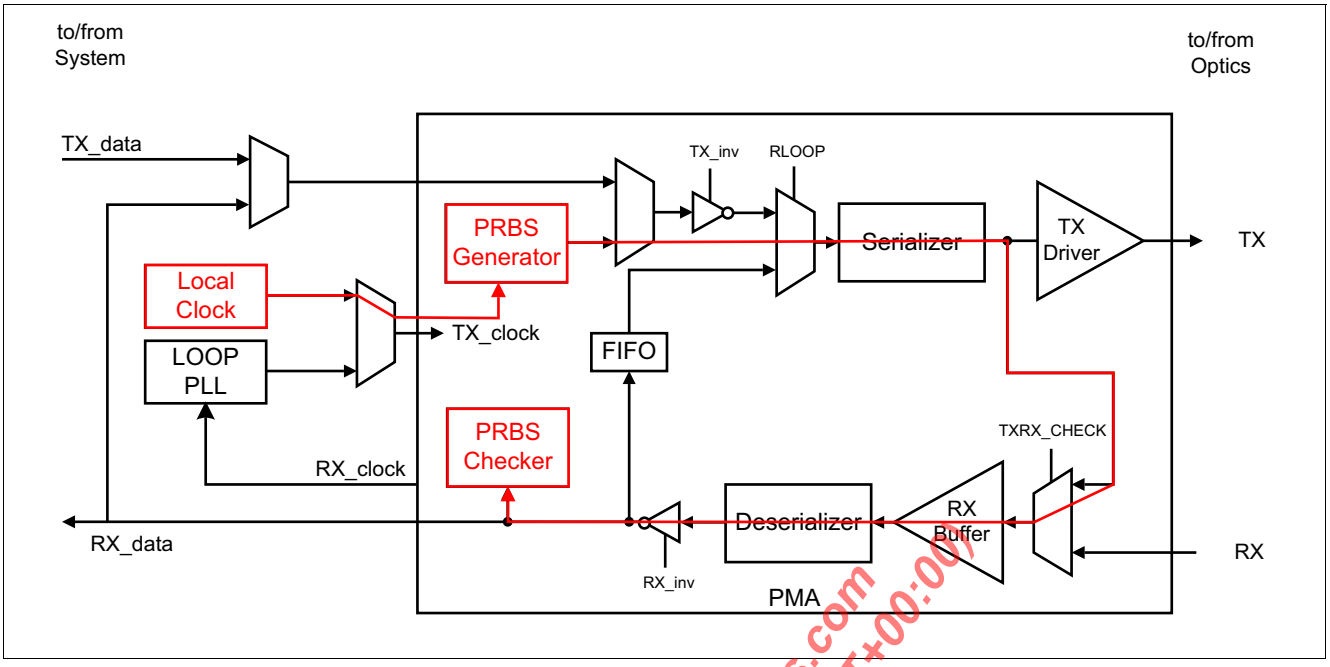


Figure 168 Optical Interface Test TXRX_CHECK Mode

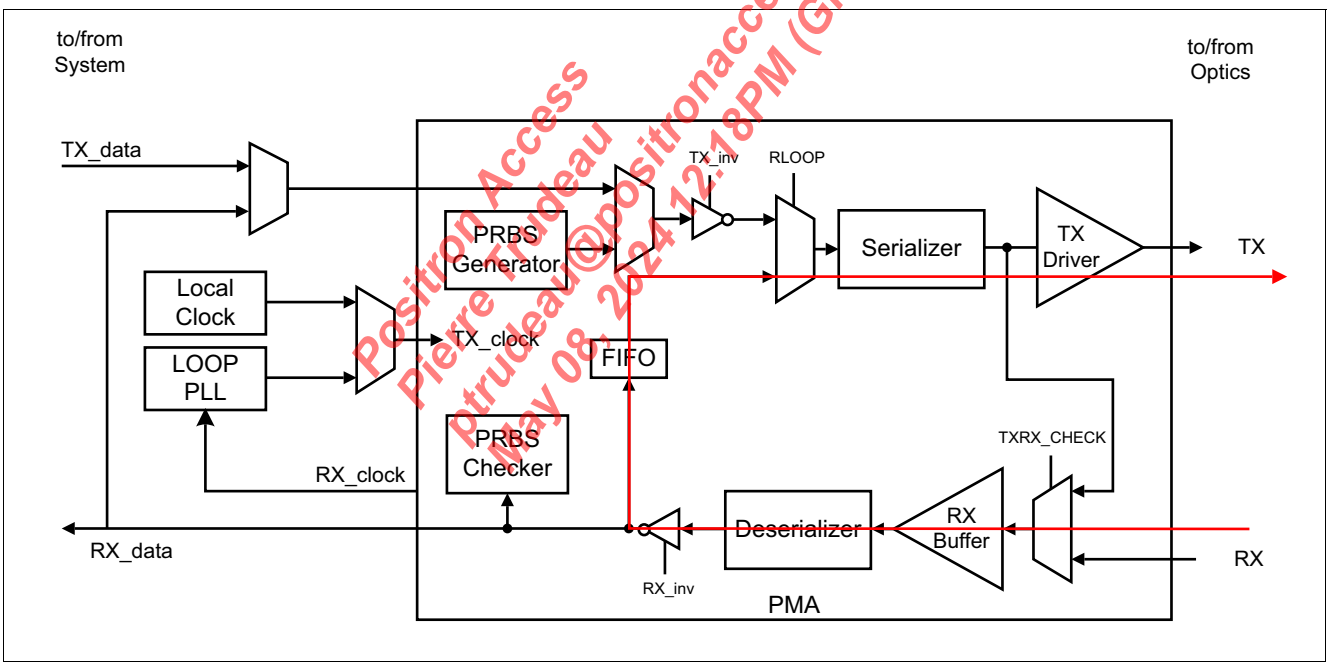


Figure 169 Optical Interface Test PMA_RLOOP Mode

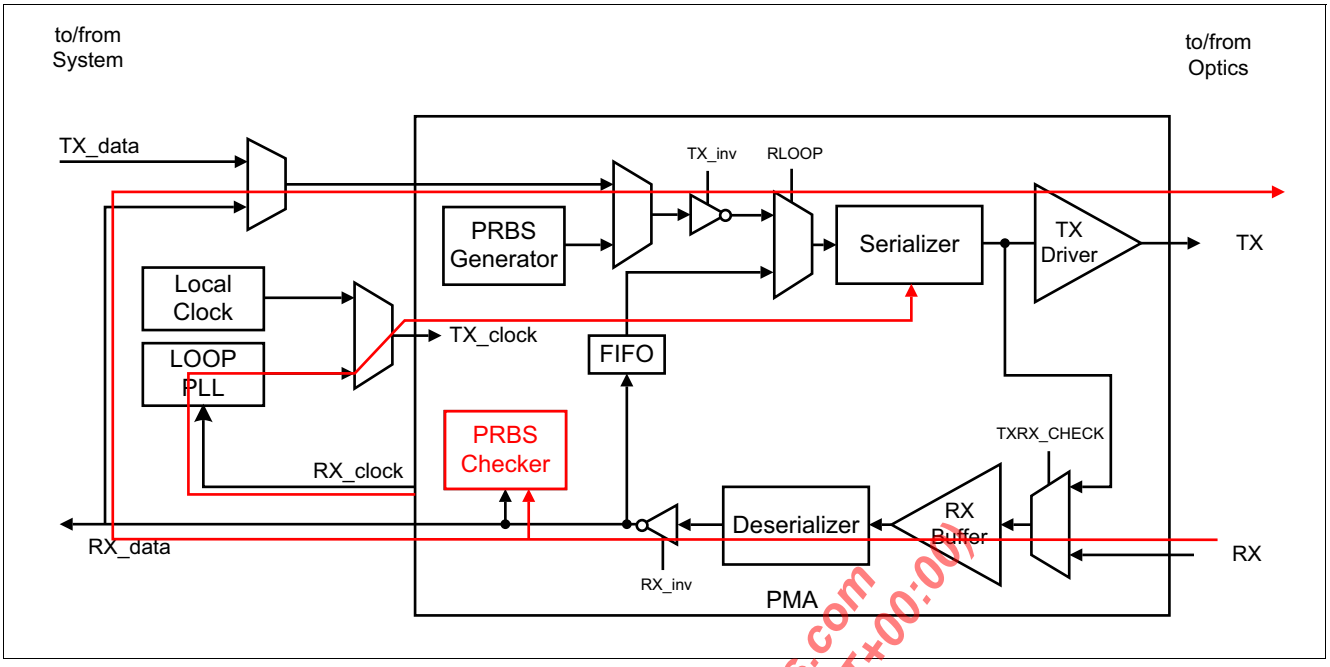


Figure 170 Optical Interface Test PON_RLOOP Mode

14.10 Power Management

Use the power management utility to check the device actual power status.

14.10.1 Power Management Help

Call the command without parameters to show the options.

```
# /opt/intel/bin/pm_util

PM_UTIL Intel Power Management Utility, Version 3.0.6
Usage:/opt/intel/bin/pm_util [options] [parameter]
options:
-h --help        Display help information
-v --version     Display pm_util version information
-a --all         Display all available power related system info
-t --temp        Display temperature sensor value
-r --refresh     Set refresh time[sec]; could be set in addition to -a, -t, -c, -p
-f --freq        specific frequency to be set [kHz]
-g --gov         new cpufreq governor [userspace|lantiqgov]
-L --log         Use together with functions -t
                 to write corresponding values to file
                 add Parameter to determine stop time in min (default = 60)
                 logfile stored in /tmp; filename(s): [ts]_log_<timestamp>.txt
-c --cfreq       Display CPU frequency; w/o -r refresh only when a frequency change
happens
-p --pwr         Display power rail monitor values[P,V]
-P --avgpw       Display power rail average for n minutes
-z --select      Select specific rail values to be displayed.
                 Can ONLY be used with -P option!
                 If there is more than one power monitor,
                 select which one(s) should be displayed in average mode.
                 Use pm_util -p to display the available power monitors to get the
                 digit selectors in the first column.
                 Refer to the example section
-s --set         This option must be set if parameters should be changed.
                 The following parameters can be changed:
                      -f frequency
                      -g governor
                      -u regulator_name min_uV max_uV
-y --dyn         ena dynamic frequency scaling [0|1|off|on]
-u --regulator   regulator options
```

14.10.2 Power Management Status

Use the all option `-a` to check the overall status.

```
# /opt/intel/bin/pm_util -a

*********************************************************************
*                          System Power Info
*********************************************************************


Thermal Zone cpu-thermal0: Temp: 69.7 C  trip0: 115.0 C passive  trip1: 125.0 C
critical
Thermal Zone cpu-thermal1: Temp: 70.8 C  trip0: 115.0 C passive  trip1: 125.0 C
critical

No Regulators found

CPU 0:
 available governors: conservative, userspace, powersave, performance
 available frequency steps: 50.0 MHz, 200 MHz, 400 MHz, 600 MHz
 current CPU frequency: 600 MHz
cur governor: performance
 cpufreq stats: 50.0 MHz:0.00%, 200 MHz:0.00%, 400 MHz:0.00%, 600 MHz:100.00%

CPU 1:
 available governors: conservative, userspace, powersave, performance
 available frequency steps: 50.0 MHz, 200 MHz, 400 MHz, 600 MHz
 current CPU frequency: 600 MHz
cur governor: performance
 cpufreq stats: 50.0 MHz:0.00%, 200 MHz:0.00%, 400 MHz:0.00%, 600 MHz:100.00%

CPU 2:
 available governors: conservative, userspace, powersave, performance
 available frequency steps: 50.0 MHz, 200 MHz, 400 MHz, 600 MHz
 current CPU frequency: 600 MHz
cur governor: performance
 cpufreq stats: 50.0 MHz:0.00%, 200 MHz:0.00%, 400 MHz:0.00%, 600 MHz:100.00%

CPU 3:
 available governors: conservative, userspace, powersave, performance
 available frequency steps: 50.0 MHz, 200 MHz, 400 MHz, 600 MHz
 current CPU frequency: 600 MHz
cur governor: performance
 cpufreq stats: 50.0 MHz:0.00%, 200 MHz:0.00%, 400 MHz:0.00%, 600 MHz:100.00%

*********************************************************************
```

14.11 Clock System Status

Print this status information to check the clock system and its individual clock rates.

```
:/# cat /sys/kernel/debug/clk/clk_summary
   clock                         enable_cnt  prepare_cnt        rate   accuracy   phase
----------------------------------------------------------------------------------------
 slic                                     1            1     8192000          0  0
 cbphy1                                   0            0           0          0  0
 cbphy0                                   1            1           0          0  0
 g_asl                                    0            0           0          0  0
 g_ashif                                  0            0           0          0  0
 g_adma                                   0            0           0          0  0
 g_aspi                                   0            0           0          0  0
 g_pd1                                    0            0           0          0  0
 g_msi1                                   0            0           0          0  0
 g_ctrl1                                  0            0           0          0  0
 g_pd0                                    0            0           0          0  0
 g_msi0                                   0            0           0          0  0
 g_ctrl0                                  0            0           0          0  0
 g_ddr                                    0            0           0          0  0
 g_switch                                 1            1           0          0  0
 g_dma3                                   0            0           0          0  0
 g_pp                                     1            1           0          0  0
 g_tdm                                    0            0           0          0  0
 g_mpe                                    0            0           0          0  0
 g_scpu                                   0            0           0          0  0
 g_secpt                                  0            0           0          0  0
 g_uart1                                  0            0           0          0  0
 g_uart0                                  2            2           0          0  0
 g_gptc2                                  0            0           0          0  0
 g_gptc1                                  0            0           0          0  0
 g_gptc0                                  1            1           0          0  0
 g_sso                                    1            1           0          0  0
 g_cqem                                   1            1           0          0  0
 g_qspi                                   1            1           0          0  0
 g_spi0                                   1            1           0          0  0
 g_spi1                                   1            1           0          0  0
 g_i2c2                                   0            0           0          0  0
 g_i2c1                                   0            0           0          0  0
 g_i2c0                                   1            1           0          0  0
 g_dma0                                   1            1           0          0  0
 g_code                                   1            1           0          0  0
 g_xbar7                                  0            0           0          0  0
 g_xbar1                                  0            0           0          0  0
 g_xbar0                                  0            0           0          0  0
 afeclk                                   1            1        8000          0  0
 osc                                      4            4    40000000          0  0
    ljpll5                                0            0  4294967295          0  0
       pon_ip                             0            0  4294967295          0  0
       pon_phy                            0            0  4294967295          0  0
```

```
      g_aon                0          0  4294967295          0 0
         g_swref           0          0  4294967295          0 0
ljpll4                     0          0  1250000000          0 0
   pcie                    0          0   100000000          0 0
ljpll3                     4          4  2500000000          0 0
   dd_ptp                  3          3   500000000          0 0
   dd_pool                 1          1  1250000000          0 0
      pondef               2          2   625000000          0 0
         g_pon             1          1   625000000          0 0
   cbphy                   1          1   156250000          0 0
   gphy                    1          1    40000000          0 0
pll2                       0          0  1600000000          0 0
   ddr                     0          0   400000000          0 0
pll1                       0          0  1572864000          0 0
   voice_if_clk            0          0    49152000          0 0
   dsp                     0          0   393216000          0 0
pll0b                      2          2  2000000000          0 0
   ngi                     2          2   500000000          0 0
   ssx4                    1          1   200000000          0 0
   sw                      0          0   666666667          0 0
pll0a                      1          1  2400000000          0 0
   qspi                    1          1   100000000          0 0
   cpu                     0          0   800000000          0 0
earlycpu                   1          1   800000000          0 0
```

14.12 Debug U-Boot

14.12.1 U-Boot Variables

To check the U-Boot variable contents from the Linux command line, use:

```
# fw_printenv <variable_name>
```

For example, to check the U-Boot version:

```
# fw_printenv ver
ver=U-Boot 2016.07-INTEL-v-3.1.234 (…)
```

These U-Boot variables are essential to run a PON system:

- `active_bank`
- `commit_bank`
- `img_validA`
- `img_validB`
- `img_versionA`
- `img_versionB`
- `ethaddr`
- `ipaddr`
- `bootdelay`

Attention: When the value of bootdelay is set to 0, the U-Boot command line cannot be entered. By default, it is set to 5 (seconds). For security reasons, it is recommended to be set to 0 for productive delivery.

Note: There are several "legacy" U-Boot environment variables that are not used anymore, please ignore and don't change them, for example: ddrconfig, firmware_gphy_firmware, wlanconfig, f_dectconfig_addr, f_dectconfig_size.

14.12.2 Update the Bootcore Binary

Depending on the file name and directory path on the TFTP server, check and adapt the variable `bootcore` to update the bootcore image. Use this to start the installation:

```
run update_bootcore
save
```

The save command ensures the version is stored into the environment and visible from Linux (`fw_printenv`).

When changes have been made to the memory partitioning, call:

```
run reset_sysconfig
```

14.12.3 Reset the U-Boot Environment

To solve compatibility issues with new U-Boot images, use this to reset the environment:

```
env default -f -a
```

After this, the non-default settings must be restored, such as `ethaddr`, `ipaddr` or other customized values.

Use `save` to save the changed settings and make them permanent.

To ensure the Linux overlay file system is cleared, run `firstboot` from the Linux CLI and then `reboot`.

14.12.4 Flash Overlay Partition Reset

Under certain circumstances, it is required to reset the flash overlay partition to make a new image run correctly. Run the `firstboot` command from the command line, followed by a reboot.

When this is unsuccessful, reset the UBI flash partition from the U-Boot prompt using:

```
run reset_sysconfig
```

14.12.5 Watchdog Timer

By default, the hardware watchdog timer is enabled in the U-Boot code and controlled by these compile options:

```
CONFIG_PRX300_WDT=y
CONFIG_CMD_WATCHDOG=y
```

When the watchdog timer is present in the U-Boot image, use these to control it from the U-Boot command line:

```
wdt stop
wdt start
```

14.13 VoIP Debugging

This chapter provides hints to the debugging of the voice system (VoIP). Using the described commands requires a debug image that includes the `tapidemo` and `tapi_daemon`. These steps are required to build a voice-enabled debug image on a Linux system:

1. Copy the SDL file to a local folder and unpack it.

2. Enter the resulting top folder and run the installation script: `./install.sh`

3. Change to `ugw_sw` and execute the preparation script `ugw_sw/ugw-prepare-all.sh`

4. Copy these tarballs to `/ugw_sw/openwrt/dl`:

```
gpon_voicedemo-2.0.0.tar.gz
lib_dti-1.3.0.tar.gz
libsrtp2-2.2.0-94ac00d5ac6409e3f6409e4a5edfcdbdaa7fdabe.tar.gz
libsrtp-1.6.0-2b091a4fb21c9b06fc5d4b5763bdaec97542fcd7.tar.gz
lib_tapi_country-1.0.5.0.tar.gz
lib_tapi_nlt_cpe-0.6.1.0.tar.gz
pjproject-2.8.tar.bz2
tapi_cli-3.3.0.0.tar.gz
tapi_sip_ua-0.7.2.tar.gz
tapidemo-5.10.8.0.tar.gz
uClibc++-0.2.5.tar.xz
voice_usr_demo-1.0.14.tar.gz
```

5. The file `ugw-8.x_master-10gpon-1.14.0-voice-tools.tgz` must be copied to the installation folder (where the `install.sh` file is located) and unpacked there (tar xf <filename>, adding some files to the feeds). Then the command `./ugw_sw/ugw-prepare-all.sh -i` must be executed.

6. Change to `openwrt` and execute `./scripts/ltq-change-environment.sh switch`. Change the software model to be built.

7. These settings must be selected in the `make menuconfig` menu:
- MaxLinear | Voice Subsystem | voice-sipua-slic (PACKAGE_voice-sipua-slic)
- MaxLinear | Voice Subsystem | voice-tapidemo-slic (PACKAGE_voice-tapidemo-slic)
- MaxLinear | Voice Subsystem | voice-tapi-cli (PACKAGE_voice-tapi-cli)

8. Run the make process: `make V=s`

Once the image is available and loaded to the target system, check:

- `tapidemo -d 1 -q -i xx.xx.xx.xx`
  – Find the IP address xx.xx.xx.xx of the `eth0_0_1_lct` interface using `ifconfig`.
- `cd /lib/firmware`
- `ln -s /lib/firmware/voice/xrx500_bbd_etsi_3_10.bin xrx500_bbd_fxs.bin`
- `sipua -n eth0_0_1_lct -i xx.xx.xx.xx -d4`
  – Use the IP address that is found by using “ifconfig”.
- tapi_daemon &

– This starts the TAPI daemon.
If the board is found and initialized some startup sequence such as this can be tested:

```
tapi_pipe TAPI_EventShow 1
tapi_pipe TAPI_EventBufferState 1

tapi_pipe TAPI_DeviceStop 0
tapi_pipe TAPI_DeviceStart 0

tapi_pipe TAPI_BBD_Download 0 0
tapi_pipe TAPI_BBD_Download 0 1

tapi_pipe TAPI_LineFeedSet 0 0 4
tapi_pipe TAPI_LineFeedSet 0 1 4
tapi_pipe TAPI_LineFeedSet 0 0 2
tapi_pipe TAPI_LineFeedSet 0 1 2
```

14.14 Debug Internal Process Communication

Internal processes use the ubus method of OpenWrt for communication. To debug this, there is a command line tool available.

Use the command `ubus list` to list all processes that are involved, for example:

```
# ubus list
log
network
network.device
network.interface
network.interface.ip6host1
network.interface.ip6host2
network.interface.iphost1
network.interface.iphost2
network.interface.lct
network.interface.loopback
network.wireless
service
system
```

To get more detailed information there is a *verbose* mode, for example:

```
# ubus -v list
'log' @27bf847e
        "read":{"lines":"Integer","stream":"Boolean","oneshot":"Boolean"}
        "write":{"event":"String"}
'network' @9c30795a
        "restart":{}
        "reload":{}
        "add_host_route":{"target":"String","v6":"Boolean","interface":"String"}
        "get_proto_handlens":{}
        "add_dynamic":{"name":"String"}
'network.device' @6fd4c819
        "status":{"name":"String"}
        "set_alias":{"alias":"Array","device":"String"}
        "set_state":{"name":"String","defer":"Boolean"}
'network.interface' @0b8a7bf1
        "up":{}
        "down":{}
        "renew":{}
        "status":{}
        "prepare":{}
        "dump":{}
        "add_device":{"name":"String","link-ext":"Boolean"}
        "remove_device":{"name":"String","link-ext":"Boolean"}
        "notify_proto":{}
        "remove":{}
        "set_data":{}
'network.interface.ip6host1' @d4f4cc67
        "up":{}
        "down":{}
        "renew":{}
        "status":{}
```

```
        "prepare":{}
        "dump":{}
        "add_device":{"name":"String","link-ext":"Boolean"}
        "remove_device":{"name":"String","link-ext":"Boolean"}
        "notify_proto":{}
        "remove":{}
        "set_data":{}
'network.interface.ip6host2' @7d6aff1a
        "up":{}
        "down":{}
        "renew":{}
        "status":{}
        "prepare":{}
        "dump":{}
        "add_device":{"name":"String","link-ext":"Boolean"}
        "remove_device":{"name":"String","link-ext":"Boolean"}
        "notify_proto":{}
        "remove":{}
        "set_data":{}
'network.interface.iphost1' @fd357993
        "up":{}
        "down":{}
        "renew":{}
        "status":{}
        "prepare":{}
        "dump":{}
        "add_device":{"name":"String","link-ext":"Boolean"}
        "remove_device":{"name":"String","link-ext":"Boolean"}
        "notify_proto":{}
        "remove":{}
        "set_data":{}
'network.interface.iphost2' @c5190d4a
        "up":{}
        "down":{}
        "renew":{}
        "status":{}
        "prepare":{}
        "dump":{}
        "add_device":{"name":"String","link-ext":"Boolean"}
        "remove_device":{"name":"String","link-ext":"Boolean"}
        "notify_proto":{}
        "remove":{}
        "set_data":{}
'network.interface.lct' @69d32e57
        "up":{}
        "down":{}
        "renew":{}
        "status":{}
        "prepare":{}
        "dump":{}
        "add_device":{"name":"String","link-ext":"Boolean"}
```

```
        "remove_device":{"name":"String","link-ext":"Boolean"}
        "notify_proto":{}
        "remove":{}
        "set_data":{}
'network.interface.loopback' @775b05f4
        "up":{}
        "down":{}
        "renew":{}
        "status":{}
        "prepare":{}
        "dump":{}
        "add_device":{"name":"String","link-ext":"Boolean"}
        "remove_device":{"name":"String","link-ext":"Boolean"}
        "notify_proto":{}
        "remove":{}
        "set_data":{}
'network.wireless' @efeaa5b9
        "up":{}
        "down":{}
        "status":{}
        "notify":{}
        "get_validate":{}
'service' @8d838bcb

"set":{"name":"String","script":"String","instances":"Table","triggers":"Array","v
alidate":"Array","autostart":"Boolean","data":"Table"}

"add":{"name":"String","script":"String","instances":"Table","triggers":"Array","v
alidate":"Array","autostart":"Boolean","data":"Table"}
        "list":{"name":"String","verbose":"Boolean"}
        "delete":{"name":"String","instance":"String"}
        "signal":{"name":"String","instance":"String","signal":"Integer"}
        "update_start":{"name":"String"}
        "update_complete":{"name":"String"}
        "event":{"type":"String","data":"Table"}
        "validate":{"package":"String","type":"String","service":"String"}
        "get_data":{"name":"String","instance":"String","type":"String"}
        "state":{"spawn":"Boolean","name":"String"}
'system' @7cdb8741
        "board":{}
        "info":{}
        "reboot":{}
        "upgrade":{}
        "done":{}
        "upgradenoreboot":{}
        "ubootvars":{}

"ubootset":{"commit_bank":"Boolean","img_validA":"Boolean","img_validB":"Boolean",
"img_activate":"Boolean"}
```

```
"watchdog":{"frequency":"Integer","timeout":"Integer","magicclose":"Boolean","stop
":"Boolean"}
        "signal":{"pid":"Integer","signum":"Integer"}
        "validate_firmware_image":{"path":"String"}

"sysupgrade":{"path":"String","force":"Boolean","backup":"String","prefix":"String
","command":"String","options":"Table"}
        "factoryreset":{"value":"String"}
        "hostname":{"value":"String"}
```

To listen to ubus events use

```
ubus listen &
```

For more options check

```
ubus help
Usage: ubus [<options>] <command> [arguments...]
Options:
 -s <socket>:           Set the unix domain socket to connect to
 -t <timeout>:          Set the timeout (in seconds) for a command to complete
 -S:                    Use simplified output (for scripts)
 -v:                    More verbose output
 -m <type>:             (for monitor): include a specific message type
                        (can be used more than once)
 -M <r|t>               (for monitor): only capture received or transmitted traffic

Commands:
 - list [<path>]                        List objects
 - call <path> <method> [<message>]     Call an object method
 - listen [<path>...]                   Listen for events
 - send <type> [<message>]              Send an event
 - wait_for <object> [<object>...]      Wait for multiple objects to appear on ubus
 - monitor                              Monitor ubus traffic
```

14.15 External Debugger Usage

It is possible to connect an external debugger to the EJTAG port for software debugging. The standard MIPS debug functions are supported. The debugger traces are stored in a section of the on-chip buffer. From the MIPS point of view, this is implemented outside the MIPS subsystem and as such is an external trace memory, although it is physically located on the same silicon.

14.15.1 Software Debug Pin Configuration

Table 94 shows the pins used to connect the external debugger.

Table 94 Software Debug Interface

External Signal Name	Debug Signal Name	Function
JT_SEL	–	Enable the EJTAG interface.
TCK	EJ_TCK	EJTAG interface
TDI	EJ_TDI	
TDO	EJ_TDO	
TMS	EJ_TMS	
TRSTN	EJ_TRSTN	

14.15.2 GNU Debugger Quick Start

This is a quick guide to get started with the GNU debugger (GDB).

14.15.2.1 Prerequisites

These tools are required:

- `gdbserver` on the target board
- `gdb` on the host PC, optionally `insight-gdb` for GUI frontend

14.15.2.2 Preparing the Board

To install the GDB server on the board, select `Advanced configuration options/Toolchain options/Build gdb` from the OpenWrt menu config. To overcome compatibility issues with different versions/targets of GDB on the host and GDB server on the platform, it is recommended to build GDB with OpenWrt. Debugging symbols must also be enabled:

- Use the debugging info option `CONFIG_DEBUG` to enable the Global build settings/Compile packages to attach debugging information to the generated binaries.
- Select the Advanced configuration options (for developer)/Target Options, then remove any optimization flags (-Os, -O1, -O2, etc) from the Advanced configuration options (for developers)/Target Options/Target Optimizations. This selection is optional and is used when there are problems.

Because adding debugging information enlarges the binaries size, it is recommended to load the root file system via NFS.

14.15.2.3 Starting a Program under GDB

Start `gdbserver` on the board with:

```
# gdbserver <boadr-ip>:1000 <binary-path> <argument 1>... <argument N>
```

This starts the GDB server and prepares a given binary to run. The binary on the board is stripped, symbols are loaded only on the host side. Start the GDB on the host with:

```
$ mips-openwrt-linux-gdb <binary-path>
```

Use this command to connect to the remote target:

```
(gdb) target remote <board-ip>:1000
```

Use this command to support the `pthreads` library:

```
(gdb) handle SIG32 pass noprint nostop
```

14.15.2.4 Working with Breakpoints

Set a breakpoint using:

```
(gdb) break <function-name>
(gdb) break <filename>:<line-number>
```

For example:

```
(gdb) break strlen
(gdb) break foo.c:99
```

Use this command to list all breakpoints:

```
(gdb) info breakpoints
```

Remove a breakpoint:

```
(gdb) delete <breakpoint-number>
```

where `<breakpoint-number>` is a breakpoint number from the `info breakpoints` command output.

Use these commands to enable or disable breakpoints:

```
(gdb) enable <breakpoint-number>
(gdb) disable <breakpoint-number>
```

14.15.2.5 Step-by-Step Debugging

When the program terminates on the breakpoint, use these commands to execute instructions.

Execute a single line of code, when the instruction is a function call not passed into it:

```
(gdb) next
```

Execute a single line of code, when the instruction is a function call passed into it:

```
(gdb) step
```

Continue the application execution until the next breakpoint:

```
(gdb) continue
```

Run until the current function returns:

```
(gdb) finish
```

Print a call stack:

```
(gdb) backtrace
```

Being at a breakpoint, the GDB shows where it stopped. To print several lines before and after the current breakpoint, use:

```
(gdb) list
```

To print the next 10 lines, retype `list`. To see the previous lines of code, use `list -`.

To examine variable values, type:

```
(gdb) print <variable>
```

Use `help print` to check the Print options.

To set a new value to a variable, use:

```
(gdb) set <variable> = <value>
```

To stop the program execution, press Ctrl-C in the `gdbserver` prompt and the program breaks.

To reduce typing, use short commands (`n` instead of `next`, `s` instead of `step`, etc).

Note: Through this document, long commands are used for clarity.

14.15.2.6 Stack Frames

Each function in the backtrace has its own stack frame.

Use this command to obtain info about current stack frame:

```
(gdb) info frame
```

Use this command to switch to a different stack frame:

```
(gdb) frame <number>
```

where `<number>` is a stack frame number from the `backtrace` command.

To print all local variables, use:

```
(gdb) info locals
```

To print function arguments, use:

```
(gdb) info args
```

14.15.2.7 Working with Threads

Use this command to list all threads:

```
(gdb) info threads
```

Use this command to switch to a given thread:

```
(gdb) thread <thread-id>
```

where `<thread-id>` is a thread ID from the `info threads` command output.

14.15.2.8 Memory Access

It is possible to read or modify arbitrary memory data.

Use this command to examine memory:

```
(gdb) x/<format> <address>
```

where `<format>` is a count followed by a format letter and a size letter and `<address>` is a symbol name (variable) or memory address.

Use `help x` to see all available options.

14.15.3 Manual Software Update

The software is manually updated from the U-Boot command line. This requires:

- The serial interface (UART) is active in the U-Boot and connected to a PC running a serial interface console application.
  The default serial interface speed is 115200 Bd (8 data bits, 1 stop bit). This is modifiable through the U-Boot environment variable `baudrate`.
- An Ethernet connection is made to a system that provides a TFTP server. The server must hold the target software image to be installed.

To enter the U-Boot, hit a key while this message is shown and the time counts down:

```
Hit any key to stop autoboot:
```

Change the U-Boot environment variable `bootdelay` to adjust the countdown time. The default value is 5 s.

When on the U-Boot prompt, make sure these environment variables are adapted to the development environment:

- `bootfile`: Must be the name of the software binary to be installed.
- `netmask`: Must match the system configuration to which the device is connected.
- `serverip`: The IPv4 address of the TFTP server.

To check the connection to the TFTP server, use `ping $serverip`. This results in a response of:

```
Using prx300-eth device
host xx.xx.xx.xx is alive
```

When not, check the connection and IP address/netmask settings.

To start the software download and installation, use:

```
run update_fullimage
```

This results in a printout similar to:

```
PRX300 # run update_fullimage
ubi0: attaching mtd1
Ecc correction detected: 01
ubi0: scanning is finished
ubi0: attached mtd1 (name "mtd=6", size 108 MiB)
ubi0: PEB size: 131072 bytes (128 KiB), LEB size: 126976 bytes
ubi0: min./max. I/O unit sizes: 2048/2048, sub-page size 2048
ubi0: VID header offset: 2048 (aligned 2048), data offset: 4096
ubi0: good PEBs: 864, bad PEBs: 0, corrupted PEBs: 0
ubi0: user volume: 7, internal volumes: 1, max. volumes count: 128
ubi0: max/mean erase counter: 53/37, WL threshold: 4096, image sequence number: 0
ubi0: available PEBs: 351, total reserved PEBs: 513, PEBs reserved for bad PEB
handling: 40
Using prx300-eth device
TFTP from server xx.xx.xx.xx; our IP address is yy.yy.yy.yy
Filename 'lede-intel_mips-prx300-PRX321_SFU_QSPI_PON-squashfs-fullimage.img'.
Load address: 0x82000000
Loading: ##################################################################
         ##################################################################
         […]
         ##################################################################
         ##################################
         1.7 MiB/s
done
Bytes transferred = 12840216 (c3ed18 hex)
```

```
Image contains header with name [PON_x1.13.0.6978]
Remove UBI volume kernelB (id 0)
Creating dynamic volume kernelB of size 2367008
Erasing NAND...
Erasing at 0x120000 -- 100% complete.
Writing to NAND... OK
Image contains header with name [MIPS 4Kec Bootcore]
Remove UBI volume bootcoreB (id 5)
Creating dynamic volume bootcoreB of size 5400896
Erasing redundant NAND...
Erasing at 0x160000 -- 100% complete.
Writing to redundant NAND... OK
Image contains header with name [UGW RootFS]
Remove UBI volume rootfsB (id 1)
Creating dynamic volume rootfsB of size 5072176
Erasing NAND...
Erasing at 0x120000 -- 100% complete.
Writing to NAND... OK
Image contains header with name [PON_x1.13.0.6978]
Remove UBI volume kernelA (id 2)
Creating dynamic volume kernelA of size 2367008
Erasing redundant NAND...
Erasing at 0x160000 -- 100% complete.
Writing to redundant NAND... OK
Image contains header with name [MIPS 4Kec Bootcore]
Remove UBI volume bootcoreA (id 6)
Creating dynamic volume bootcoreA of size 5400896
Erasing NAND...
Erasing at 0x120000 -- 100% complete.
Writing to NAND... OK
Image contains header with name [UGW RootFS]
Remove UBI volume rootfsA (id 3)
Creating dynamic volume rootfsA of size 5072176
Erasing redundant NAND...
Erasing at 0x160000 -- 100% complete.
Writing to redundant NAND... OK
Remove UBI volume rootfs_data (id 4)
Creating dynamic volume rootfs_data of size 33554432
```


To activate the new software image, reboot the system by either

- typing `reset`,
- pressing the reset button, or
- power cycling the ONU.

15 Security Functions

This chapter describes the security related functionality.

To achieve a high level of system security, a number of measures are taken to prevent unprivileged access to the system and to inhibit modifications resulting in an unstable or even unusable system. These security functions must also ensure that the ONU does not disturb other ONUs on the same PON fiber branch (OLT port).

Attention: The default settings of the software apply all security settings to a high level. Additional measures may be required to achieve the security level required by the target application. This and any reduction of the security level is under control and responsibility of the OEM software implementer.

As a summary, these aspects are subject to specific security handling.

- **OTP Memory**
  Local credentials and security settings are stored on-chip in a one time programmable (OTP) memory. This ensures the settings are physically fixed and cannot be changed after they have been programmed. The trusted execution platform (TEP) has exclusive physical access to this memory.
- **System boot process**
  This ensures that only software images containing a valid security key are loaded.
  The Secure Boot option is strictly recommended for productive systems.
  An insecure boot mode option is possible, but this must be used only for development, debugging, or production purpose in a safe environment.
- **Protected storage of the U-Boot environment**
  The U-Boot environment is protected to ensure a secure boot process.
  Essential U-Boot environment variables are stored in an encrypted format.
- **Secure software update**
  This ensures that only authorized software images are updated to the flash memory.
  The loading of any malicious code is prevented even before the next system boot is triggered.
- **Software roll-back prevention**
  Internal hardware flags in the OTP are used to block downloading of older outdated software revisions, as marked by their roll-back ID.
- **Secure storage of PON-related credentials**
  Various credentials must be handled to establish a connection between the ONU and an OLT. These must be stored locally in flash and are encrypted. This prevents modification by a malicious user who tries to get unauthorized access or break the connection to the OLT.
  A command line tool is provided to enter these values during the production process.
- **Secure PON firmware download**
  This method prevents an unauthorized PON firmware binary to be loaded into the PON IP hardware. This is an additional security level to protect the PON access. A malicious PON firmware may potentially disturb the function of other ONUs on the same OLT port.
- **Secure debug mode**
  All debug functions are disabled by default, including the serial UART interfaces. This prevents malicious users to get access to the operating system command line either through the serial port (ASC/UART) or through the local Ethernet port(s) (UNI) by means of Ethernet-/IP-based protocols.
  For system debugging, it is possible to enable the secure debug mode but a debug key is required to unlock it. As the OLT is viewed as a trusted device in the operator safe environment, the OLT, through the PPTP LCT UNI managed entity, is able to enable the Ethernet access (for example SSH) through the UNI port(s).

For additional UGW reference documentation, refer to **[16]** and **[22]**.

15.1 Secure Image Build Options

The TEP processor software is always part of the default system build and used for the boot sequence.

Attention: To enable the security features, a minimum external data memory size of 512 MB is required.

15.1.1 Debug Access to Registers and Memory

By default, the `io` command which gives access to memory locations and hardware registers is not compiled in. To enable it for debug images, the configuration must be changed to:

```
CONFIG_KERNEL_DEVMEM=y
```

15.1.2 OTP Memory Programming

For more information, refer to **[16]**, chapter 7.7.

The `otp.bin` file contains the data written to the OTP memory for provisioning.

A new file is created by executing the `gen-keys-otp.sh` script without any arguments. This deletes all existing keys and generates new keys using the Linux random pool. This results in the output of:

- `blkey.bin`
  - 256 bit asymmetric ECDSA public key, generated from `ecdsa_keypair_A.der`, added to the `otp.bin`
- `CRkey.bin`
  - 256 bit symmetric AES key, also added to the `otp.bin`
- `ecdsa_keypair_A.der`
  - 256 Bit asymmetric ECDSA private key, **not** added to the `otp.bin`
- `nvmimage.vhx`
  - `otp.bin` data with some explaining comments
- `otp.bin`
  - Data to be written to the internal OTP memory region
- `otp_bin.h`
  - OTP data

*Note: By default, no secure debug key is added to the `otp.bin` file. To generate a key, see **Section 15.7.1**.*

Secure Debug Unlock Bit Configuration

To change the unlock bits controlled by the secure debug mode, the bin/rootkey.py file must be modified.

For example:

```
--- a/bin/rootkey.py
+++ b/bin/rootkey.py
@@ -55,7 +55,7 @@ def
NvmImage(PUKey,PublicKey,DbgKey_1,DbgKey_2,DbgKey_3,Opt_DbgKey_4):
             nvm.add(NvmRsaAuthKeyObject_SecureDebug(5, DbgKey_3, '0200'))
     if Opt_DbgKey_4 is not None:
         if len(Opt_DbgKey_4) > 1:
-            nvm.add(NvmRsaAuthKeyObject_SecureDebug(4, Opt_DbgKey_4, '0200'))
+            nvm.add(NvmRsaAuthKeyObject_SecureDebug(4, Opt_DbgKey_4, 'ff00'))
     nvm.add(nvmtool.NvmTailData())
     nvm.out_vhx('nvmimage.vhx')
```

15.2 PON Credential Programming

Credential values required for PON operation are stored in file referred to as CPDO, cryptographically protected data object, in the flash memory. This ensures that operation critical key material or other values are stored in an encrypted format and cannot be read as clear text from the flash memory.

The data contents must be created once during ONU production. Most of the content does not change over the ONU lifetime but some values may be changed, for example, the PON registration ID.

The key to encrypt and decrypt the CPDO is stored in the OTP. The TEP has access to the OTP and performs encryption and decryption of the data.

15.3 Secure Boot

For a description of the secure boot process, refer to **[16]**.

Figure 171 shows the secure boot as selected in the U-Boot configuration.

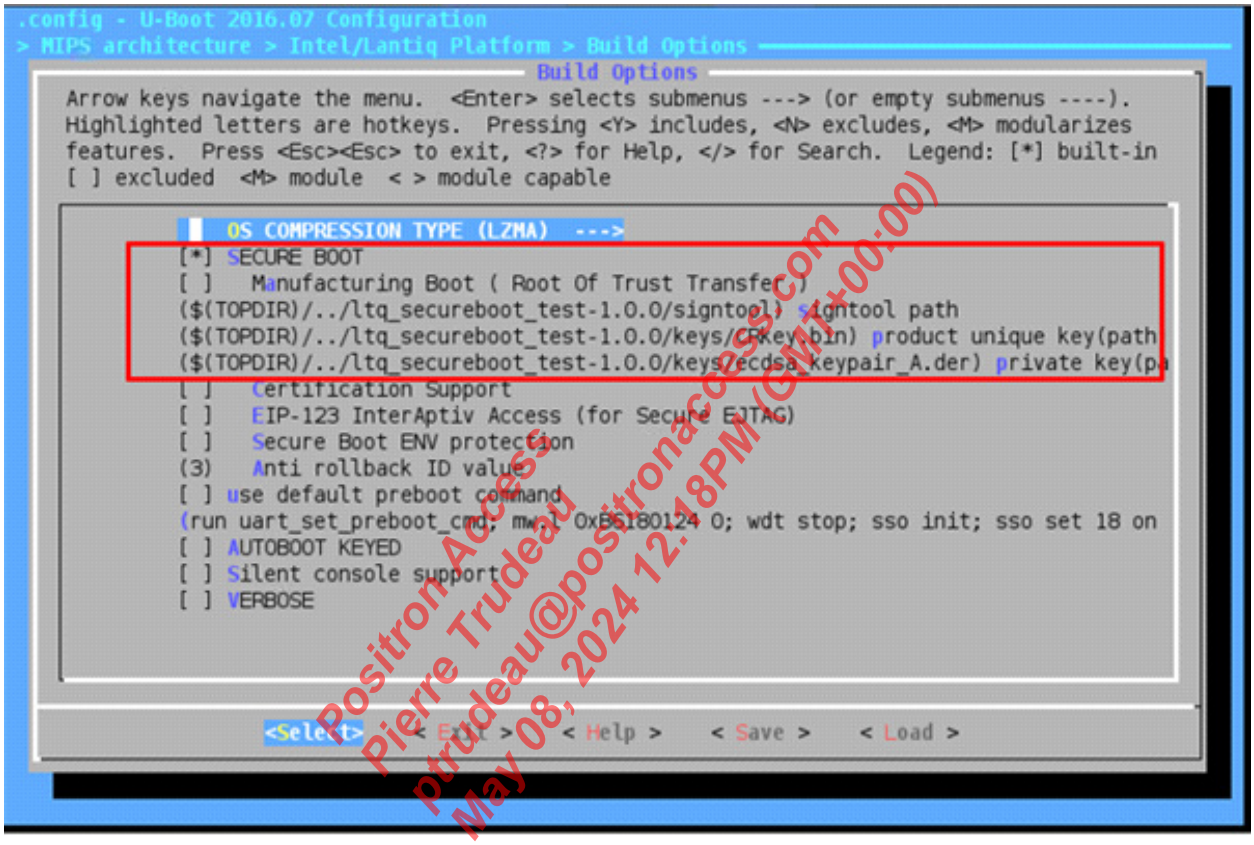


Figure 171 Secure Boot Configuration

15.4 Protected U-Boot Environment

The protection of U-Boot environment variables is available as of the U-Boot version 3.1.189 and later.
By default, all environment variables are protected when protection is enabled (see **Figure 171**).
Most variables of the default environment are reset to their respective default values during the boot process.

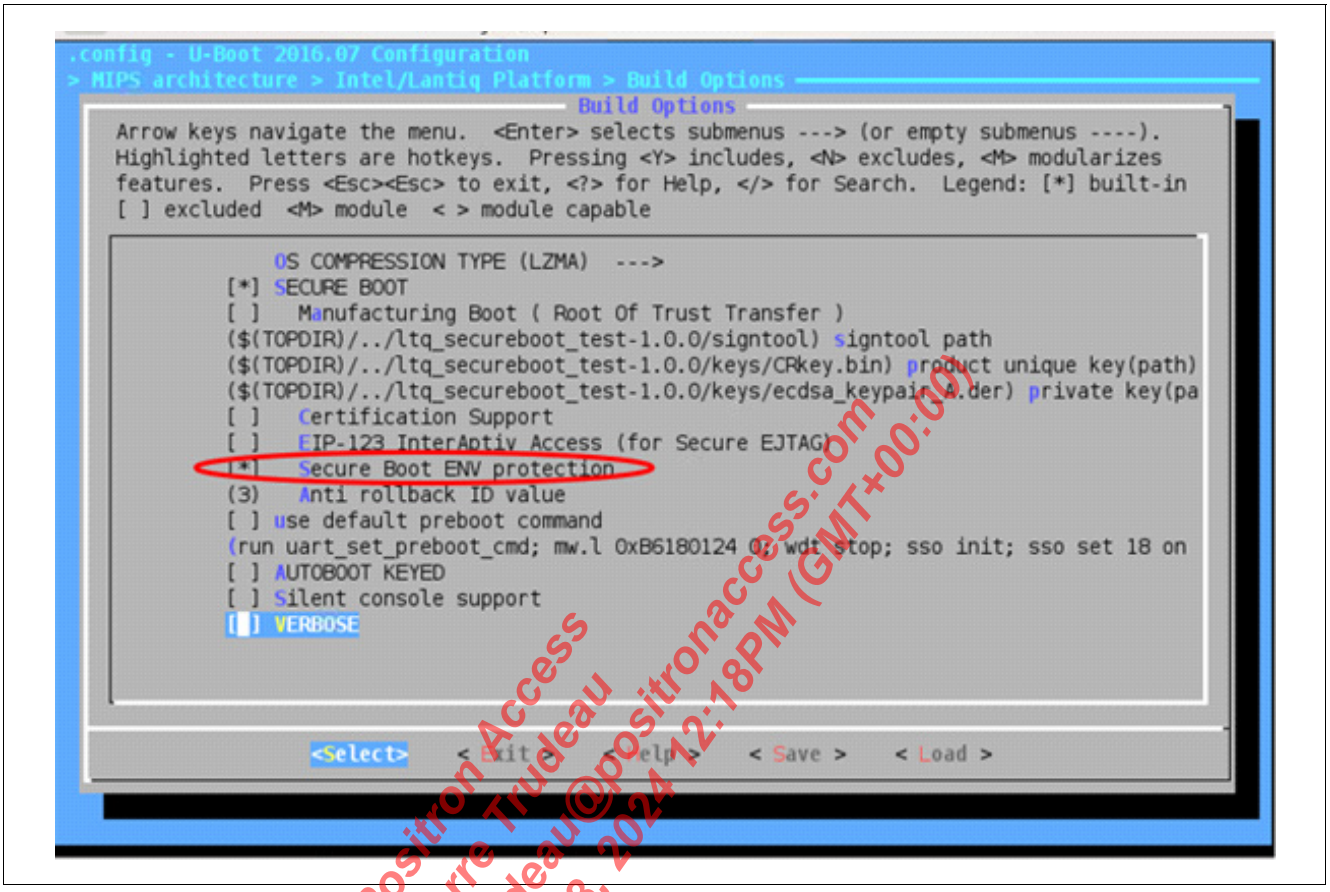


Figure 172 Secure Boot Configuration

To exclude specific environment variables from the protection, the U-Boot environment creation script must be modified. **Figure 173** shows an example that excludes `<var1>` and `<var2>`.

The script is located within the U-Boot patches at `<u-boot-base-path>/scripts_platform/mk_envimg.sh`.

```
OUTPUT=$1

ADDRHEX='sed -e' '/default_environment/!d' u-boot.sym | awk '{print substr($1,length($1)-4,5)}''

ADDR='echo $((16#$ADDRHEX))'

LENHEX='sed -e '/default_environment/!d' u-boot.sym | awk '{print $2}''

LEN='echo $((16#$LENHEX))'

dd if=u-boot.bin of=env.tmp ibs=1 skip$ADDR count=$LEN

# secure boot env remove protection of some variables:
# 1. Exclude <var1>  from protection:
sed -i.bak 's/<var1>=\([^\x0]\+\)\x0//' env.tmp
# 1. Exclude <var2>  from protection:
sed -i.bak 's/<var2>=\([^\x0]\+\)\x0//' env.tmp
```

Figure 173 Excluding U-Boot Environment Variables from Protection

15.5 Secure PON Firmware Loading

During the system startup, the PON firmware must be loaded into the PON IP hardware module. The PON firmware binary is part of the encrypted software image. The secure PON IP firmware loading process ensures that the PON IP is loaded correctly and that the firmware binary is loaded only once during the system initialization phase. The load process is performed by the PON mailbox driver (`pon_mbox_drv`). The `pon_mbox_drv` asks the TEP to enable PON firmware download. When the firmware download is completed, any subsequent attempt to download another (possibly malicious) PON firmware binary is inhibited.

This security function can be disabled for debug images by a build option (compile switch).

To enable the PON firmware download process, the secure firmware download key must be present in the CPDO memory (see **Section 15.2**). Follow these steps to provide this:

1. From the U-Boot prompt, enforce the secure PON firmware download:
   ```
   PRX300 # mw.l be2007d8 00000002 1
   ```
2. Create a debug-capable software image, based on the PRX300_DEBUG build model.
   Modify the file `ugw_sw/feeds/ugw/net/system_service/sse_fapi/config/Config.in` in this way:
   ```
   --- a/sse_fapi/config/Config.in
   +++ b/sse_fapi/config/Config.in
   @@ -1,6 +1,5 @@
    config EIP123_SUPPORT
           bool 'Enable EIP123 Support'
   -       depends on INTEL_MIPS_KEYS_PROVISION
           default y
           help
             Enable EIP123 SUPPORT
   ```

Storing the secure PON firmware download key into the CPDO part of the flash memory requires a debug image. This image must be built by using these options:

```
CONFIG_EIP123_SUPPORT=y
CONFIG_PACKAGE_sse_fapi-sse_test_app=y
```

Before building the image, copy the generated secure PON firmware download key into the `openwrt/files` folder:

```
/openwrt$ mkdir files/
/openwrt$ cp /local/<user>/pon/prx300_otp/dbg-3-der.private files/dbg-3-der.private
```

Build the image and run it on the target system.

To store the key, use:

```
:/# sse_test_app store ./dbg-3.private buf len = 2459
```

When completed, build the image which uses the secure PON firmware download key by setting these build options:

```
CONFIG_EIP123_SUPPORT=y
CONFIG_PACKAGE_sse_fapi-sse_test_app=y
CONFIG_PACKAGE_kmod-pon-mbox-drv_SKIP_TEP=n
```

Build this image and load it to the target system. The secure PON firmware download key from the CPDO is now used to authenticate the PON firmware download process.

15.6 Software Roll-back Prevention

The secure header contains the roll-back ID. When this value is larger than the value stored in the OTP memory, the OTP value is updated to the higher value.

The roll-back prevention feature allows older software revisions to be locked out when important updates have been made, for example to improve the system security. The roll-back ID is assigned to an image during the secure image signing process as described in [16], chapter 6. During the boot process, the TEP compares the image roll-back ID with the value stored in the OTP and continues booting only with an image ID equal or larger.

The roll-back ID stored in the OTP consists of a 64-bit value. Each time the roll-back ID is increased, one additional bit is set. This limits the number of overall roll-back ID increments to 64.

The available bits are separated into independent parts related to FSB/U-Boot, TEP, and Linux.

The valid values are 0x0, 0x1, 0x3, 0x7, 0xF, 0x1F, and so on until all bits in the OTP roll-back memory space are set. When all the available physical bit positions have been programmed, no further roll-back protection is possible.

15.6.1 Roll-back Prevention under OMCI Control

The software update process under OMCI control, through the OLT, uses a dual-image method to ensure that a valid backup image is available as a fall-back, when the new image does not operate properly.

When incrementing the roll-back ID, this fall-back to the earlier image is not possible as it is blocked by the roll-back prevention mechanism.

To handle this properly, it is recommended to complete these steps:

1. Download the new image without roll-back ID increment.
2. Check that the new image works properly.
3. Download the same image again with roll-back ID increment.

15.7 Secure Debug Mode

When the secure debug mode is used, local system debugging is disabled by default. This prevents local access to the system by potentially malicious ONU end users.

For laboratory use, production configuration and calibration, or for field debugging by trusted maintenance personnel, the local access can be re-enabled.

For more details and a general description, refer to **[16]**.

15.7.1 Secure Debug Key Generation

To use the secure debug mode, the secure debug key must be generated. The generation flow is as follows.

1. Use OpenSSL to generate a private RSA key with a key length of 3072 bit in PEM format:
   `openssl genrsa -out dbg-priv.pem 3072`
2. Convert the key into the DER format:
   `openssl rsa -in dbg-priv.pem -inform PEM -out dbg-priv.der -outform DER`

This results in these files:

```
dbg-priv.der
dbg-priv.pem
dbg-pub.der
```

The key must be programmed into the OTP memory, as described in **Section 15.1.2**.

15.7.2 EJTAG Debugging

The EJTAG debug interface is intended for use during software development. This interface is physically disabled when the secure debug mode is activated.

To reactivate the EJTAG interface, follow the procedure described in **[16]**.

15.7.3 Local Serial Interface

For enhanced system security, it is recommended not to provide physical access to the device UART interface signals. Even when they are physically present, the usage is disabled by default.

When the ONU hardware is equipped with a serial interface connector, the operating system command line access can be enabled on this interface. For the EJTAG interface, follow the steps described in **[16]**.

15.7.4 Local Ethernet Interface

The local Ethernet interfaces are by default blocked for remote access to the operating system command line function by using a firewall. This access can be re-enabled through the UART interface or remotely by the OLT to allow system debugging by stopping the firewall (`/etc/init.d/firewall stop`). This is also required to re enable `tftp` for testing.

It is strictly recommended to use an encrypted protocol such as SSH and enforce user authentication, by password or certificate.

Note: Insecure protocols such as `telnet` *are not part of the software delivery and must not be implemented.*

The local debug access is controlled by the OLT through the OMCI management protocol, using the PPTP LCT UNI managed entity and its administrative state attribute.

Literature References

[1] 10G PON Chipset PRX120 Data Sheet Rev. 2.5

[2] 10G PON Chipset PRX126 Data Sheet Rev. 3.6

[3] 10G PON Chipset PRX321 Data Sheet Rev. 3.5

[4] 10G PON Chipset GNX321 (GNX321B0BC/GNX321B1BC) Data Sheet Rev. 3.4

[5] 10G PON Development Kit EASY PRX126 REF BOARD V2.2.3 (SFP+) HW6.1.03 Hardware Design Guide Rev. 1.1

[6] 10G PON Development Kit EASY PRX321 REF BOARD V1.10.2 HW6.1.02 Hardware Design Guide Rev. 1.2

[7] 10G PON Chipset System Overview Rev. 3.0

[8] 10G PON Chipset 10G PON Subsystem Software Overview Rev. 2.0

[9] 10G PON Chipset System Package 1.22.1 Release Notes

[10] 10G PON Subsystem Optical Timing Calibration Application Note Rev. 2.0

[11] 10G PON Chipset Packet Buffer Application Note Rev. 1.0

[12] 10G PON Chipset DDR Selection Application Note Rev. 4.0

[13] 10G PON Chipset Secure Boot Build Application Note Rev. 1.3

[14] 10G PON Chipset SerDes Interfaces Application Note Rev. 2.0

[15] Ethernet Network Connection GPY API V2.7.1.2

[16] Secure Boot Application Note Rev. 2.0

[17] Ethernet Network Connection GPY215 (GPY215B1VI, GPY215C0VI) Data Sheet Rev. 1.4

[18] Universal Gateway Software 8.x Software Overview Rev. 2.0

[19] Universal Gateway Software 8.7.1.10 for URX Series Getting Started Rev. 1.0

[20] Universal Gateway Software 8.x QoS Programmer's Reference Rev. 1.1

[21] Universal Gateway Software 8.7.1.10 for URX Series Components Sheet Rev. 1.0

[22] Universal Gateway Software 8.x Debug Application Note Rev. 1.0

[23] URDK Software 8.8.0.100 for AnyWAN™ Broadband SoCs Release Notes Rev. 1.0

[24] Device tree specifications, **https://www.devicetree.org**

[25] Doxygen tool for code documentation, **http://www.doxygen.nl**

[26] iproute2 utilities, **https://wiki.linuxfoundation.org/networking/iproute2**

[27] Linux kernel device tree documentation, **https://www.kernel.org/doc/Documentation/devicetree**

[28] Netlink library, **http://www.infradead.org/~tgr/libnl/**

[29] Policing action in tc, **http://man7.org/linux/man-pages/man8/tc-police.8.html**

[30] prplOS at GitLab, **https://gitlab.com/prpl-foundation/prplos**

[31] prplOS Wiki, **https://confluence.prplfoundation.org/display/PRPLWRT/prplOS**

[32] prplOS PON ubus Data Model, **https://confluence.prplfoundation.org/display/PRPLWRT/Ubus+PON+DM**

[33] prplOS XPON Manager, **https://confluence.prplfoundation.org/pages/viewpage.action?spaceKey=PRPLWRT&title=XPON+Manager**

[34] Comparing prplOS with OpenWRT, **https://confluence.prplfoundation.org/display/PRPLWRT/Comparing+prplOS+with+OpenWRT**

[35] RDK-B, **https://wiki.rdkcentral.com/display/RDK/RDK-B+Overview**

[36] RDK-B OpenEmbedded build system, **https://en.wikipedia.org/wiki/OpenEmbedded**

[37] Token Bucket Filter, **http://man7.org/linux/man-pages/man8/tc-tbf.8.html**

[38] ubus, OpenWrt Micro Bus Architecture, **https://openwrt.org/docs/techref/ubus**

[39] Verizon, OpenOMCI Specification V2.2, **https://github.com/vz-etech**

Attention: Refer to the latest revisions of the documents.

Note: Some of the referenced Internet links may require registration.

Standards References

[40] BBF TR-156, Using GPON Access in the context of TR-101

[41] BBF TP-247, G-PON & XG-PON & XGS-PON ONU Conformance Test Plan

[42] BBF TR-280, ITU-PON in the context of TR-178

[43] IEEE 802.1 Suite of Standards for Ethernet networking and bridging

[44] IEEE 802.1AE MAC Security

[45] IEEE Std 1588-2008 Precision Clock Synchronization Protocol

[46] IETF RFC 4115

[47] IETF RFC 2687

[48] IETF RFC 2697

[49] IETF RFC 2698

[50] IETF RFC 3339

[51] ISO/IEC/IEEE 8802-3 Standard for Ethernet

[52] ITU-T G.984 series: Gigabit-capable passive optical networks (G-PON), including:
G.984.1: General characteristics
G.984.2: Physical media dependent (PMD) layer specification
G.984.3: Transmission convergence layer specification

[53] ITU-T G.986: Digital sections and digital line system – Optical line systems for local and access networks

[54] ITU-T G.987 series: 10-Gigabit-capable passive optical network (XG-PON) systems, including:
G.987.1: General requirements
G.987.2: Physical media dependent (PMD) layer specification
G.987.3: Transmission convergence (TC) layer specification

[55] ITU-T G.9807.1: 10-Gigabit-capable symmetric passive optical network (XGS-PON)

[56] ITU-T G.988: ONU Management and control interface (OMCI) specification

[57] ITU-T G.989: 40-Gigabit-capable passive optical networks (NG-PON2), including:
G.989.1: General Requirements
G.989.2: Physical media dependent (PMD) layer specification
G.989.3: Transmission convergence (TC) layer specification

[58] ITU-T G.8262 Timing Characteristics of Synchronous Equipment Slave Clock

[59] ITU-T G.8262.1 Timing Characteristics of enhanced Synchronous Equipment Slave Clock

[60] ITU-T G.8271/Y.1366, Packet over Transport aspects – Synchronization, quality and availability targets, 2016-07

[61] ITU-T G.8273.2 Timing characteristics of telecom boundary clocks and telecom time slave clocks for use with full timing support from the network

[62] ITU-T G.8273.4 Timing characteristics of telecom boundary clocks and telecom time slave clocks for use with partial timing support from the network

[63] ITU-T G.8275.1 Precision time protocol telecom profile for phase/time synchronization with full timing support from the network

[64] ITU-T G.8275.2 Precision time protocol telecom profile for phase/time synchronization with partial timing support from the network

[65] NIST Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, 05-2005

[66] NIST FIPS 197, Specification for the Advanced Encryption Standard (AES), 11-2001

[67] SFF-8472, Management Interface for SFP+, Rev 12.3, 07-2018

[68] UM10204 I²C-bus Specification and User Manual, NXP Semiconductors, 10-2012

Attention: Refer to the latest revisions and optional amendments of the documents.

Terminology

1DM	One-way Delay Measurement
1PPS	One Pulse per Second signal
A	
ANI	Access Network Interface, WAN-side (**PON**-side) network interface of a PON **ONU**
AON	Active Optical Network, also known as “Active Ethernet”
API	Application Programming Interface
ARC	Alarm Reporting Control
AUX	Auxiliary interface or signal
B	
BC	Boundary Clock
BITS	Building Integrated Timing Supply
BMCA	Best Master Clock Algorithm
BOSA	Bidirectional Optical Subassembly
BP	Bridge Port
BPF	Berkeley Packet Filter
C	
CA	Classifier Action (Linux QoS)
CLI	Command Line Interface, Linux console to enter manual commands for debugging
CPU	Central Processing Unit (microprocessor core)
CSD	Configuration Storage Daemon
CTP	Connection Termination Point, logical interface where a physical connection is terminated
D	
DDMI	Digital Diagnostics Monitoring Interface for pluggable optical transceivers (SFP), as defined by SFF-8472
DDR	Double Data Rate
DHCP	Dynamic Host Configuration Protocol
DMM	Delay Measurement Message
DMR	Delay Measurement Reply
DNS	Domain Name Server
DownMEP	Downstream Management End Point
DPLL	Digital Phase-locked Loop
DRAM	Dynamic Random Access Memory
DSCP	Differentiated Services Code Point
DTS	Device Tree Source
DTSI	Device Tree Source Include file
E	

EEC	Ethernet Embedded Clock
eEEC	Enhanced Ethernet Embedded Clock
EJTAG	Enhanced JTAG, uses the JTAG physical interface to connect software debug equipment
ETH	Ethernet (abbreviation)
F	
FAPI	Functional **API**
G	
GEM	G-PON Encapsulation Method
GPHY	Gigabit-capable Ethernet PHY
GPIO	General Purpose Input/Output
GPS	Global Positioning System, provides the global time in a dedicated format, with a fixed offset (leap seconds) to the **TAI** time scale.
H	
HAL	Hardware Abstraction Layer
HGU	Home Gateway Unit, a PON ONU that includes home gateway functionality
I	
I^2C or IIC	Inter-Integrated Circuit Interface as defined by NXP UM10204; also referenced as two-wire bus
I^2C M/S	Inter-Integrated Circuit Interface (Master/Slave)
ICMP	Internet Control Message Protocol
ID	IDentifier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol, control protocol for IPv4 multicast forwarding configuration
IIR	Infinite Impulse Response
IP	Internet Protocol or Intellectual Property
IPNPM	Internet Protocol Network Performance Management
ITU	International Telecommunication Union
J	
JSON	JavaScript Object Notation
JTAG	Joint Test Action Group (here: Test Interface)
K	
KiB	1 KiB = 1024 byte
L	
LAN	Local Area Network, user-side network of a PON ONU
LLDP	Link Layer Discovery Protocol
LMM	Loss Measurement Message
LMR	Loss Measurement Reply
M	
MAC	Media Access Control

MDIO	Management Data Input/Output
MDU	Multi Dwelling Unit
ME	Managed Entity
MEP	Management End Point
MIP	Management Intermediate Point
MLD	Multicast Listener Discovery Protocol, control protocol for IPv6 multicast forwarding configuration
MOOC	Master Only Ordinary Clock
MSA	Multi-Source Agreement
MTU	Multi Tenant Unit
N	
netconf	Network Configuration Protocol defined by the IETF
netdev	Network Device (Linux)
NDP	Neighbor Discovery Protocol (used by IPv6)
NMEA	National Marine Electronics Association, synonym for a serial output format of GPS information as defined by this association.
O	
OAM	Operation, Administration, and Maintenance
OC	Ordinary Clock
OLT	Optical Line Terminal
OMCC	ONU Management and Control Channel
OMCI	ONU Management and Control Interface
ONU	Optical Network Unit
OOC-PSD	Out-Of-Channel optical Power Spectral Density
OpenWrt	Linux-based open-source framework for home gateway applications
OTP	One-time Programmable memory, used to hold system identifiers which must not be changed
P	
PCE	Packet Classification Engine (with parser, classifier, and metering capability)
PCIe	Peripheral Component Interconnect express
PCS	Physical Coding Sublayer
PDU	Packet Data Unit
PHC	PTP Hardware Clock
PHY	Physical Layer Circuit, typically an interface to external components
PLL	Phase Locked Loop
PLOAM	Physical Layer **OAM**
PMD	Physical Media Dependent Sublayer
PON	Passive Optical Network
PPS	Pulse Per Second, also known as 1PPS; synchronous time reference signal
ppb	Parts per Billion
PTP	Precision Time Protocol (as defined by ITU-T 1588)

prplOS	Linux-based framework for home gateway applications maintained by the prpl Foundation
PSK	Pre-Shared Key, secret which is known only to both ends of an encrypted channel
Q	
qdisc	Queuing Discipline (Linux QoS)
QoS	Quality of Service
QSPI	Quad Serial Peripheral Interface
R	
RDK-B	Reference Design Kit for Broadband
RSTP	Rapid Spanning Tree Protocol (newer version of **STP**)
S	
SDK	Software Development Kit
SFP	Small Form Factor Pluggable Module
SFU	Single Family Unit
SGMII	Serial Gigabit Media Independent Interface
SID	Stream Identifier
SoC	System on Chip
SOOC	Slave Only Ordinary Clock
STP	Spanning Tree Protocol
SW	Software
SyncE	Synchronous Ethernet
T	
TAI	International Atomic Time, maintained by the Bureau International des Poids et Mesures. TAI time is not adjusted for leap seconds and is ahead of **GPS** time by 19 seconds.
T-BC	Telecom Boundary Clock
TC	Transmission Convergence (PON protocol layer)
TC	Thread Context (Linux)
TC	Traffic Control (Linux QoS)
TCA	Threshold Crossing Alert (used by OMCI to trigger alarms)
TCI	Transaction Correlation Identifier (used by the OMCI protocol) or Tag Control Information (used in VLAN tags)
T-CONT	Transmission Container
TCXO	Temperature Compensated Crystal Oscillator
TDM PON	Time Division Multiplex Passive Optical Network (single-wavelength PON, such as G-PON, XG-PON, or XGS-PON)
ToD	Time of Day
trTCM	Two-rate Three Color Marker
TWDM PON	Time and Wavelength Division Multiplex Passive Optical Network (multi-wavelength PON, such as NG-PON2)
U	
UART	Universal Asynchronous Receiver/Transmitter
UDS	UNIX Domain Socket

UCI	Unified Configuration Interface
ubus	Inter-process communication protocol used by OpenWrt and prplOS
UGW	Universal Gateway (Software)
UNI	User Network Interface
UpMEP	Upstream Management End Point
UXSGMII	Universal 10 Gigabit Serial Media Independent Interface
V	
VLAN	Virtual Local Area Network
VoIP	Voice over IP
W	
WDM	Wavelength Division Multiplex
WFQ	Weighted Fair Queuing
WNE-PSD	While Not Enabled optical Power Spectral Density
WRED	Weighted Random Early Detection
X	
XFI	XFP Interface
XTAL	Crystal Oscillator

Index

Numerics

A

B

C

D

E

F

G

H

I

T

U

V

W